1971

The design, development and translation of general purpose software for the P3C aircraft's digital computer.

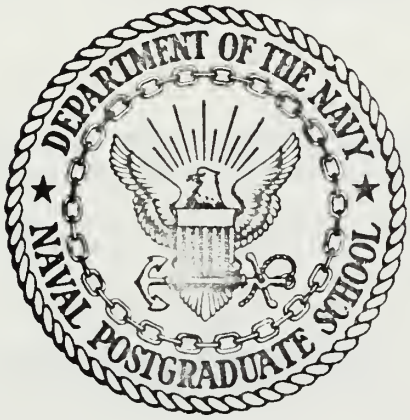
Ray, Dennis Edward.

Monterey, California. Naval Postgraduate School

THE DESIGN, DEVELOPMENT AND TRANSLATION OF GENERAL PURPOSE SOFTWARE FOR THE P3C AIRCRAFT'S DIGITAL COMPUTER

Dennis Edward Ray

NAVAL POSTGRADUATE SCHOOL
Monterey, California



THESIS

THE DESIGN, DEVELOPMENT AND TRANSLATION
OF
GENERAL PURPOSE SOFTWARE
FOR THE
P3C AIRCRAFT'S DIGITAL COMPUTER

by

Dennis Edward Ray

Thesis Advisor: G. H. Syms

December 1971

Approved for public release; distribution unlimited.

The Design, Development and Translation of General Purpose Software for the P3C Aircraft's Digital Computer

by

Dennis Edward Ray
Lieutenant, United States Navy
B.S., United States Naval Academy, 1964

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1971

ABSTRACT

The Navy has incorporated a modified UNIVAC 1830-A (CP 901 or ASQ/114) "mini" digital computer into its P3C aircraft. This ASQ/114 computer system is presently used only during aircraft testing and flying. In the near future, fifty or more of these digital systems will be operational and will sit virtually idle about 40% of the time. Hence, this project was undertaken to improve the computer utilization, and to provide the individual squadrons with an administrative computer capability.

Six specific tasks leading to the implementation of a general purpose operating system have been undertaken: a feasibility study, development of a CP 901 translator, design and development of an assembler, design study of the bootstrapping technique, design of a FORTRAN compiler, and the design of a control operating system.

The documentation of these six tasks is intended to aid in the development of the final system.

TABLE OF CONTENTS

LIST OF TABLES

I. INTRODUCTION

The ASQ/114 special purpose digital computer system is an integral, perhaps the most important, part of the Navy's P3C Orion ASW aircraft. This aircraft is in fact a computerized integration of crew operations, aircraft weapons and support systems, sensors, and automatic navigation and steering. This integration of airborne sensors with a digital computer is termed the A-NEW concept. The major advantage of the A-NEW system lies in the computer's programmed ability to accomplish the myriads of detailed tasks in order to make logical decisions faster and more accurately than can be done manually.

This system incorporates computer-controlled displays at the TACCO (TACtical CoOordination Officer), the Pilot and the non-acoustic operator stations for the presentation of aircraft, sonobuoy, and visual contact information. These displays are continuously updated, and with the aid of a relatively simple set of function switches, each operator controls and modifies his display to satisfy his station requirements. Additional displays at the TACCO and the NAV/COM operator stations enable them to view and modify numerous tables containing data regarding the status of equipment, weapons, search store devices, audio systems, and navigation. The computer has released the Navigator of virtually all of the mental integration and hand calculation normally required in the solving of the navigation problems. It regularly receives navigational data to update the aircraft displays; but also continuously monitors the active navigation system operation and automatically selects an alternate system in the event of a failure or malfunction. The computer can display upon request either a latitude/longitude or a grid co-ordinate position of

the aircraft, sonobuoys, contacts, or other selected positions. Thus by specifying preselected positions, the computer can provide precise steering courses to that selected point. The computer further manages the stockpile of weapons and search devices, etc., on board and automatically presents TACCO with alternatives for improper or unfulfillable selections.

The system as installed in the P3C Orion provides for the automatic recording of essential mission data on magnetic tape. This dayfile is taken as snapshot views of the mission profile at preselected regular intervals in the flight or whenever a significant event occurs.

A section of the ASQ/114's central processor, namely the bootstrap memory, is composed of 512 words of non-alterable code for use in loading the system's operating program and also in the automatic restart of the computer. This bootstrap memory is electrically unalterable and provides the restoration capability of the computer to save its environment, the entire aircraft status, and the display system. Thus minor computer interruptions cause only a temporary delay in the system operation, as a virtually instantaneous restore function can be executed which automatically reloads the operating system and the last status of the aircraft allowing normal system operation to continue without loss of information.

The heart of the A-NEW system is then the CP 901 digital computer which, under the control of the operational program, frees the aircraft operators from manually performing a number of functions. This in turn allows the operators considerably more time to concentrate on and perform their primary mission task (ASW problem).

II. RESEARCH OBJECTIVES

The primary goal of this research was based upon the satisfactory completion of a feasibility study of utilizing the special purpose digital computer system (ASQ/114) as a general purpose system (section IV). In essence then the main goal was to provide the ASQ/114 system users with the software to utilize the hardware in a general purpose digital system and, thus improve upon the system utilization. System utilization is a computer factor defined as the ratio of a system's actual processing use time to the possible time. For the ASQ/114 system this factor was conservatively estimated as 60%. There were many sub-goals categorized during the initial description of the problem. Of these, five have been completed; the description of which make up a substantial part of this report. These are described in sections V through IX.

The non-availability of an ASQ/114 system in the local area required the development of the CP 901 translator on the NPS IBM 360/67 (section V). Its primary purpose is to provide a testing tool for programs coded in UNIVAC machine code. The original CP 901 assembler (section VI) was programmed in IBM PL/1. It also is a tool, but is designed to be used in connection with the bootstrapping technique (section VII) to produce a CP 901 assembler written in CP 901 assembly language. This assembler can in turn be used to produce a FORTRAN compiler (section VIII) and a control operating program (section IX) as described in section VII.

By completing these five sub-goals, an individual or group associated with NPS could readily complete the installation of an ASQ/114 general operating system with a minimum of effort. Sufficient documentation and

description of these goals is provided in order to permit followup efforts to continue from this point, rather than requiring additional effort be expended in investigating system factors already considered.

III. ASQ/114 COMPUTER SYSTEM

A. ASQ/114 HARDWARE

The ASQ/114 system hardware was developed by the UNIVAC Corporation as a specially configured 65K memory UNIVAC 1830-A digital computer. The 1830-A is a miniaturized (8 cubic feet, 395 lbs.) general purpose, real-time, stored program machine that is composed of four relatively light weight modular assemblies: central processor unit, memory unit, power supply unit, and input/output unit. The CPU is an example of a typical processor configuration with a control section and an arithmetic section which together perform arithmetic and logic functions under the direction of the control section. The memory unit consists of ferrite core modular arrays of 4096 thirty bit words. Memory is overlapped between 16K word banks, with a maximum machine memory capacity of 128K. The overlap operation involves addressing any two memory units simultaneously. The power supply unit converts the input voltage, 115 VAC 400 cycles to regulated direct current. The input/output unit provides up to sixteen thirty bit parallel input or output channels.

The military modification to the 1830-A occurred primarily in the sixteen channel and interface units required for the specialized aircraft peripheral I/O units. The interfacing of all peripheral equipment is controlled in either one of the three logic units (LU-1, LU-2, and LU-3) or one of the five converter or interface units (teletype interface, DIFAR interface, synchro-digital interface connector, radar interface, and a data bank converter).

The major peripheral equipment includes two seven track magnetic tape drives, a teletype CRT/teletype display as well as various minor output

lights, panels, and CRT displays. Normal military input units include radar, navigation, doppler, hand keysets, control panel settings, various acceptance switches, and other special military input devices.

Operation in the central processor utilizes overlap memory techniques to decrease the effective instruction set execution times. In the ASQ/114 overlap is the simultaneous reading of the instruction for the next operation from one memory unit and the operand for the present instruction from another memory unit. The CPU also provides the I/O capability of overlapping memory addressing. In this case overlapping is the ability for the I/O and the program instruction set to simultaneously access different memory units. The I/O operation has priority in the case of all conflicts.

Within the CPU, the control section utilizes the U register, a thirty bit register, to hold the instruction word during execution of an operation. The function code and the various instruction designators are translated from the appropriate bits of this register. If an address modification is required before execution, the contents of the appropriate B register is added to the proper section of the U register. The B register set is a group of eight fifteen bit registers used as address modification registers or as index registers.

The P register is a fifteen bit register that holds the memory address of a computer instruction word, that of the next instruction to be loaded into the U register for execution.

The K register functions as a shift counter for all shift operations. Such operations that use this register are the arithmetic functions of multiply, divide, or square root.

Associated with the A register, the conventional accumulator, are the CPU working registers the A* and the X. A typical function of addition

would shift the contents of the A register into the A* register. The X and the A* registers would then be added in a parallel bit operation with the results being placed into the A register.

The thirty bit Q register is principally used during the multiply and divide operations. The contents of both the A and the Q register may be shifted left or right or may be combined into one sixty bit word. All shifts are accomplished by the parallel exchange network in conjunction with the A* and Q* registers. Neither the A*, Q*, nor the X register is directly addressable, as these are only used by the CPU for the exchange of data within the arithmetic section.

Each 16K memory unit contains two registers, the S and the Z. The S register contains the fourteen bit address of the location within the unit being referenced. The Z register holds the thirty bits of data being written into or read from the memory location specified by the S register.

In order to access all four memory units, it is necessary to use the "indirect" memory addressing scheme. The normal address portion of an instruction in the U register is fifteen bits, which permits access to only 32K of memory (2 to the 15th power is 32K). Thus in the direct mode of addressing bit fifteen of the address selects either unit zero or unit one and the remaining fourteen bits are passed to the selected unit's S register. In the "indirect" mode of addressing bits eleven through fifteen of the instruction's address select one of the sixteen six bit Absolute Page Registers (APR). The six bits of the selected APR are transferred to the R2 register. The upper three bits determine the memory unit to be used and the lower three buts determine the page within the unit to be addressed (pagesize is 2K). The original remaining eleven bits of the address then specify the word within the selected page.

All input/output operations are passed through the computer's I/O section, which provides for twelve input channels and sixteen output channels. These channels are assigned priority in descending order from fifteen to zero. They are further divided into four 4 channel groups with descending priority from three to zero. Only one channel in a group may be active at any time, but all four may be transmitting data simultaneously.

There are four basic methods of communication using the channels between the computer and any of the peripheral equipment or devices. Data transfers may be made with or without Monitor. With Monitor an interrupt is generated upon completion of the data transfer, so that immediate processing of the change of data can be initiated. Without Monitor the data transfer is merely completed and no specific action is taken until a program instruction is encountered directing that action. The four methods of data transfer are:

1. Input data transfer (with or without Monitor) is the normal method of transferring data from a peripheral unit to the computer's memory.
2. Output data transfer (with or without Monitor) is the normal method of transferring data from the computer's memory to a peripheral unit.
3. External function (with or without Monitor) is an output function of higher priority than a normal output transfer, and usually requires the peripheral unit to perform some action or provide some reply. External function with force is a variation of this mode that passes some form of data to a peripheral unit that is not capable of requesting the data.

4. External Interrupt is the highest priority of input data transfer and is usually used to report a significant change of status.

All data transfers are accomplished via a buffering process. This buffering allocates an area of memory into which or from which the data transfer is made. It then permits normal program execution and data transfer to proceed on a time shared basis until all of the words have been transferred.

Interrupts (internal and external) temporarily halt the executing instruction and place the attention of the CPU upon the event that generated the interrupt. Internal interrupts stem from computer power failures, program faults, memory protection violations, and I/O interrupts generated upon the completion of data transfers with Monitor. External interrupts include those generated from logic unit power failures, input contacts, weapon release switch activations, etc. All interrupts cause a jump to a memory location containing the instructions for handling the specific type of interrupt that was generated.

The inputs and outputs to and from the computer's memory consist of the data transfers and coded control signals. These control signals are specific bits of information to indicate requests or acknowledgments of the receipt of some transfer of information. Data transfers are thirty bit words transmitted over thirty parallel data lines which may carry normal thirty bit data words or coded signals. There are twelve input data points, five of which are multiplexers. Similarly, there are sixteen output stations, six of which are multiplexers. For each station there exists a special unique format word identifying the station and desired function of operation. Within these thirty bit words there exist numerous field variables dependent upon the particular station involved.

B. ASQ/114 SOFTWARE OPERATING SYSTEM

The operating program of the ASQ/114 system is a 62K program that is a pseudo simulation of the environment of the aircraft. All of the aircraft's variables are actively retained in core memory at all times. These variables are updated as necessary and on a continuing basis in order to maintain the aircraft status and system displays. At the same time the system keeps a copy of the memory on magnetic tape. This tape copy is used to recover the system and the environment in the event of a temporary computer or system malfunction. Magnetic tape transport #1 is thus dedicated to the operating system and the recovery program. On the other hand magnetic tape transport #2 is dedicated to receive the running picture of the snapshots of the events occurring during the flight. This dayfile is saved for evaluation at a later date.

The system responds to each input request, first determining the unit from which the request was made. Then it ascertains the word format of the request and breaks down the coded fields in order to respond to the request. Once the proper response has been computed, it is coded into thirty bit words of the proper format for the requesting station. The information is then transferred to the requesting peripheral unit and there it is decoded into the proper display format in order to be displayed.

The operating program is a program written in the seventy-two available basic operation codes (3652 operations with the various modifiers). These fall into the categories of arithmetic operations of add, subtract, etc., as well as jumps, shifts, repeats, comparisons, load and store functions, various I/O functions and interrupts. The basic operations codes (opcodes) are combined with the various modifiers and the

operand address into one of five possible format types. These formats are illustrated in Appendix A. The formats are variations of the bit pattern interpretations used to identify the different functional instructions and the included data of the format word. These types include two variations of a general instruction format, a special I/O format, a special non-arithmetic format, and an addressing format. These instruction formats are broken down in the CPU and its associated registers to perform the desired function.

Within the framework of the operating program numerous instruction sets have been built to perform the wide variety of complex operations required. These operations include a communications network, an external computer linking capability, the analog input linking and the thousands of record keeping tasks maintained within the system. All of these factors provide an operating aircraft with a capability superior in every way to previous versions of the aircraft, combined with a significantly improved safety factor for all of the aircraft system operators.

IV. FEASIBILITY STUDY: ASQ/114 AS A GENERAL PURPOSE SYSTEM

In evaluating the ASQ/114 system for its general purpose capabilities, knowing its special purpose application, eight specific areas were considered. These eight categories fall under the following general headings:

1. History of the system's digital components
2. Untried procedure problem areas
3. Implementation problem areas
4. Anticipated benefits
5. Operating overhead factors
6. Personnel problems
7. Digital system language
8. System design concepts

The heart of the ASQ/114 system is a modified UNIVAC 1830-A digital computer. The basic design of the 1830-A is from a long line of production models of various UNIVAC miniaturized computing systems. The immediate predecessor of the 1830-A was the Phoenix system which used a 1830 as its digital computer. Thus the digital components of the system are totally compatible with the standard UNIVAC system. The seventy-two basic machine instructions are identical with those for the standard 1830-A computer. Similarly the two magnetic tape drives of the 1830-A system are machine compatible with most UNIVAC systems as well as a few IBM systems.

In the initial testing and evaluation of the A-NEW system, a set of diagnostic programs was developed by the UNIVAC Corporation to isolate equipment errors in a minimum of time. The existence of this set of routines lends credibility in believing that a general programmer can in fact program the A-NEW system for general purpose use.

There are many pitfalls in attempting to install a new operating system, especially where a present system does not exist. In this case there are no blocks to build upon. At best the system designer is attempting to fit a software package into an existing hardware system. Although difficult, this concept is certainly a reasonable, if not the only, approach to use in the development of an operating system. Ideally the design of an operating system should be developed simultaneously with the associated hardware, but in this case the software will at best be an adaptation. As such the efficiency of the operating system will be somewhat less than that of a simultaneously developed system. However, total efficiency of the system is not of significant concern providing reasonable program execution can be achieved, especially since the intended applications would utilize normally unused computer time.

The number of P3C aircraft equipped with ASQ/114 systems in the next two years is anticipated to be about fifty. Considering that as a minimum these systems will be idle about 40% of the time and as the number of aircraft systems increases, the percentage of idleness will undoubtedly increase (due to the increased aircraft maintenance and repair time). The total weekly computer time available then is about 3500 CPU hours — a considerable amount of potential compute power.

As the number of aircraft increases, likewise the number of records, files and supply functions at a Naval Air Station will also increase but at a rapid pace. The number of administrative personnel required to manage these areas will also spiral upwards. If the idle CPU time could be channeled towards managing this area, the end result would be a more efficient flight squadron.

Assuming that an operating system (control program, higher level language compiler, and an I/O monitor) could be developed, the benefits gained by using the computer's idle time must be weighed against the operating overhead. Since the system is composed of a miniaturized computer, the operating power requirements are very minimal, less than one kilowatt. Power can be provided either directly from the aircraft's auxiliary power supply or from an external cable source readily available in the normal aircraft support facility.

The most serious hardware limitation to the use of the ASQ/114 system as a general purpose digital system is the output facilities. The aircraft's line printer is a non-impact type that uses a heat sensitive type of paper. This paper costs about eighty cents for a page containing about twenty-five lines. The cost of a single line of about 100 characters is about 3.2¢ (about 160 times higher than normal computer printer output). The TACCO's CRT could as well be used for displayed output, but it is temporary and limited by the size of the screen. The approximate number of output lines visible at one time is eighteen. The only other feasible output device for general purpose use would be either of the system's tape drives. These tape drives are UNIVAC and IBM compatible. The drawback with these devices is that some other compatible computer system must be available for reading of the taped output generated from the ASQ/114 general purpose system. This limitation is discussed further below.

Of significant importance to the installation of a general purpose operating system is the present lack of EDP trained personnel within the aviation community, especially in the field of the managment of data processing equipment. In order to adequately use the general purpose

system, it is imperative that knowledgeable personnel operate the system. It is anticipated that the minimum training required for an operator would be some background in programming coupled with three weeks of direct active training with the general purpose operating system.

In order for the system to yield the maximum usefulness throughout the range of locations of the ASQ/114 system, it was felt that since FORTRAN was probably the best known language, it should be the language implemented into the general purpose operating system. The FORTRAN compiler will be developed using the translator and the assembler (sections V and VI) along with the bootstrapping technique (section VII). The implementation programs will consist of a control program, a CP 901 assembler, and a FORTRAN compiler all written in CP 901 assembly language.

In summary then the greatest limiting factor in implementing a general purpose operating system is the limited I/O capabilities of the ASQ/114 system. The existence of the two tape drives provides the ease of handling file manipulation problems. The lack of significant output printing capability severely limits the size of a problem's printed output. Although the hardware output capability is limited, this does not preclude the use of the ASQ/114 system as a general purpose computer, especially for the many applications requiring a small amount of output. (If the system proves to be sufficiently useful and effective, then the possibility of acquiring and interfacing of a special output printer should be investigated). Also of the three possible input methods available in the present configuration (direct panel coding, keyboard typing, and magnetic tape) none are very satisfactory for long program inputs. Direct panel coding requires programming in the system's machine code which is a very long and tedious process. Keyboard typing appears to be

the most likely method of input to the system, whether for data input or for programming. Tape input would require an original tape generating program either on this system or on a compatible system. Another limitation of the hardware is the size of core memory (65K). The maximum size of any program is thus limited to something less than 65K. With judicious use of the control program to monitor the printed output as well as due concern for the limited size of a program, one can still accomplish a very wide range of computing problems. There is under development a 128K drum addition to the ASQ/114 system. The addition of this paging drum will more than double the possible program size and thus will eliminate size as a limitation of the general purpose system.

After considering the various factors, it appears that only the I/O problem is of significant concern in implementing a general purpose. By carefully programming around this limitation, many different utilizations of the ASQ/114's system idle time are feasible.

V. CP 901 TRANSLATOR

The CP 901 translator is a design and testing tool used in the development of the initial assembler. It is written in PL/1 to be run on the IBM 360/67 at NPS, and is used to verify the machine code generated by the assembler and the compiler. Thus the assembler, the FORTRAN compiler, and all programs written for this system could be written and debugged at this installation, instead of requiring an ASQ/114 system from a P3C aircraft.

The translator is a direct coding of the repertoire of basic CP 901 instructions given in appendix A to reference 1. Of particular concern were the many variations affecting the basic seventy-two operation codes. Where possible and practical, these variations were coded as separate sub-procedures. Since the maximum allowable size of an array on the IBM 360/67 is 32K, it was necessary to split the dummy memory into an upper and a lower half. Considerable difficulty was encountered during all arithmetic operations due to the octal vs. hexadecimal representation in the CP 901 vs. the IBM 360. Thus it was necessary to force the conversion of all numeric information and addresses into octal representation prior to any arithmetic operation.

The translator program begins execution with memory address 620, the initial program loading point of the CPU's bootstrap loader. The dummy U register is set to the contents of memory address 620. The various fields are then isolated and a jump to the designated "opcode" is executed. Within each of the opcode sections the function of the opcode is performed as modified by the various designators. After completion of the function, the next sequential instruction word is loaded into the

dummy U register so that the same process can then be repeated. The sequential stepping through memory continues until either a programmed jump to another address is encountered or a termination occurs. All output from the executed routine is placed on output file sysprint on the IBM 360.

The internal coding of the functions within the various opcodes is reasonably straight forward. The comments that precede each section explain the function of the section as well as the variables and function calls. Wherever possible the use of PL/1 built-in functions were utilized. The major functions so used were BOOL, BIT, BINARY, SUBSTR. All other functions or procedure calls are user-defined.

A listing of this translation routine is included after appendix D.

VI. DESIGN AND DEVELOPMENT OF IBM 360 CP 901 ASSEMBLER

One must always keep in mind that a computer only understands its own machine language as specified by the manufacturer. Programming in machine language is generally extremely laborious, because the language is usually very complex, unnatural from English, difficult to memorize, and very vulnerable to errors. These difficulties are emphatically true with the UNIVAC 1830-A machine instruction set. Although there are only seventy-two basic opcodes, the six modifiers actually provide for a total of 3652 machine instructions. Direct machine language programming is thus extremely tedious.

A basic assembly language is the next higher level of computer language. It more closely resembles English forms of expression, but is not easily recognizable as English due to the coding schemes utilized for each expression. Assembly language coding is significantly better for the programmer than machine coding; but often the programmer must write nearly as many assembly instructions as he did machine language instructions. Remember that the computer cannot execute instructions written in assembly language directly, but by means of the assembler translator, these statements are transformed into executable machine code. However, writing anything but very small programs in assembly language is still a long and tedious task. A means of reducing this task still further will be discussed in section VIII, but it requires an operational assembler for development.

To initially alleviate some of the problems of writing programs in machine language for the CP 901, an assembly language set was defined. This set is based upon an existing language and is expressed in the

Backus Naur Form (BNF) syntax and is given in appendix C. The definitions and the functions of the language statements are listed in Table I. Note that this general set does not conform directly with the machine language set, thus many assembly instructions will require multiple machine instructions. In this sense this assembler is a "macro assembler" rather than a one-for-one assembly to machine code translator.

This assembler written in PL/1 contains significantly more operators than required for the initial generation of a CP 901 assembler. These excess operators have been provided as a direct guideline for future implementation using the bootstrap technique. Some of the operators associated with arrays and procedure calls are not defined or documented, as such they are intended for reference only.

There are thirty-two labels permitted in the assembler. These are numbered sequentially from 540 - 577 octal. All program object machine code will be loaded into memory starting at address 620 octal. With this fact in mind the address of the machine instruction labelled in assembly code is stored in memory at the address of the label itself. Note that the labelled address must be stored prior to a programmed reference to that label. This can either be a forced programmed storage or may be accomplished by the assembler. To assist the assembly language programmer, the number of machine instructions issued from every assembly instruction is listed in Table I. The programmer can thus determine the memory location of pertinent machine instructions for referencing and modifications. This is accomplished by using address 620 as the base and adding to it the number of machine instructions for each assembly instruction issued. Thus for a simple program set of ADD, ADD, MUL, DIV, the first machine instruction of the divide assembly instruction would be located in memory at address 625.

TABLE I

Operation, #1 , #2	Function	#Inst
ADD , X	Add to register A the value of field X	1
SUB , X	Subtract from register A the value of field X	1
MUL , X	Multiply register A by the value X	3
DIV , X	Divide register A by the value X, dividend in A, remainder in Q register	4
EXP , X	Raise A to X power, result in A	4
LES , X	Skip the next instruction if X is less than A	3
EQL , X	Skip the next instruction if X is equal to A	3
LEQ , X	Skip the next instruction if X is less than or equal to A	1
NEQ , X	Skip the next instruction if X is not equal to A	3
GEQ , X	Skip the next instruction if X is greater than or equal to A	3
GTR , X	Skip the next instruction if X is greater than A	1
NEG , X	Skip the next instruction if X is negative	3
NOT	Compliment A	1
AND , X	If X = A Compliment Q	5
BOR , X	If X or Q = 0 Compliment A	5
ONE , X	A = X Q = 0	1
TWO , X , Y	A = X Q = Y	2
LOD , X	A = X	1
STO , X	Store A in memory (X)	1
STD , X	Store A in memory (X) set A = 0	2
XIT	Restart the program, reset all registers	4
BFN , X	Branch forward X instructions	3
BFC , X	Branch forward X instructions if A greater than zero	3
BBC , X	Branch backward X instructions if A greater than zero	3

TABLE I (Continued)

Operation, #1 , #2	Function	#Inst
BRS , X	Branch to instruction address stored in memory address (X)	3
BSC , X	Branch to instruction address stored in memory address (X) if A ≠ 0	3
NOP	Remain idle and wait for data channel	1
PRO	Procedure call (for future use)	
RTN	Return from a procedure (for future use)	
GET , X	Allocate X memory locations for an array (for future use)	
RET , X	Return saved area X (future use)	
RDV , X	Read variable into A using data channel X	3
WRV	Output A	3
WRS , X	Write string of X words stored at memory location 5 (11 words max.)	X+3
DMP , X	Output bits of A (future use)	
TAB	Future array use	
ROW	Future array use	
SCR	Future array use	
SAV	Future procedure use	
UNS	Future procedure use	
SUP , X	Future error handling call to the operating system	
LIT , X	A = X Q = X	1
EOF	Terminate execution set P = 0	1
DEL	Set A = 0	1
DUP	Set Q = A	1
XCH	Exchange A and Q	1

All of the comparison operators (LES, LEQ, GTR, etc.) require a branching instruction as the next instruction for the "false" path of execution. Note that all branching, whether forward or backward implies a knowledge of the actual machine instruction address. However, this indirect type of addressing is translated into direct instruction addressing via manipulation of the P register and the count of the number of machine instructions per assembly instruction.

Use of this assembly language implies a knowledge of the associated machine code. However, by carefully studying the assembly language operator descriptions, a competent programmer can write routines in CP 901 assembly language with a minimum of effort. Thus with this PL/1 version of the assembler at NPS, any program coding is possible on the school's IBM 360 computer. The object code produced would than be executable on the CP 901. In general very long assembly programs tend to be infeasible due to the tedious task of inputting this larger number of statements into the computer (as discussed above). In this case of a large program it would be necessary to use a higher level language translater to produce CP 901 machine coded programs.

VII. BOOTSTRAPPING TECHNIQUE

With the simple CP 901 assembler written in IBM 360 PL/1 and executing on the NPS IBM 360/67, the process of developing a complete CP 901 assembler written in CP 901 machine language through a series of executions is termed "bootstrapping".

One of the major concerns in the bootstrap development process is the absolute error free initial generation run. In order to facilitate this requirement, usually the initial generation is kept as short as possible in order to reduce the probability of logic and programming errors. This generation must also accept at least the minimum number of input instructions in order to generate a workable set of primary machine coded instructions.

Once a working assembler is generated in the primary machine language, it can be isolated as the operating assembler. This program can then be loaded into either the translation memory or the actual CP 901 system for execution. Under this execution the second source input program produces the second generation CP 901 assembler.

Consider the generation scheme as depicted in appendix B. The initial assembler generation source input is a series of instructions acceptable to the PL/1 CP 901 assembler (CP ASS to CP 901 TRANS). The output from this execution is a simple assembler written in CP 901 machine language. This coded set of instructions will then accept CP 901 assembly language programs under execution on the CP 901 system. By making incremental changes to the source program and executing the change on the previous assembler, the size of the next version of the assembler can be increased.

Although this description is a rather oversimplified version of the actual bootstrapping process, it does illustrate the principles involved. The necessary programming to achieve the initial assembler is quite substantial. Once the initial assembler has been generated, however, the step by step additions become relatively easy to install. More effort, however, is required in the testing of new additions than in the installation of the change itself.

VIII. DESIGN CRITERIA FOR A FORTRAN COMPILER

The standard electronic computer is not an electronic brain, but a simple machine that does calculations on data by obeying a sequence of instructions. This series of instructions is uniquely peculiar to each machine, its own machine language. As is amply documented, programming in machine language for anything but the most trivial of tasks is beyond the general programmer's scope of ambition. A similar argument exists for assembly language programming. Thus there is the need for a more powerful and user-oriented programming language.

In considering a higher level language for the general purpose operating system for the ASQ/114, an analysis of the following factors was required: the intended users, size of the language compiler, ease of use of the language, and the anticipated user program types. Briefly the results of that analysis are: the general purpose user of the system will generally have little if any programming experience. These users will primarily be the administrative managers within the aviation community. Their main use of the system is anticipated to be largely confined to the handling of the administrative records and files as well as in the maintenance of various supply records. As such there will probably be a mixture of computer tasks ranging from pure bookkeeping and the arithmetic functions to file handling and file maintenance. With the limited computer memory size available (65K words), the size of the language compiler and its resources must be compatible with the CP 901 hardware. Thus the general purpose system's primary user language

must be reasonably short, simple to use, and capable of handling both arithmetic and file manipulation tasks.

In order to meet these requirements, it was decided that the basic system language should be FORTRAN. FORTRAN has the arithmetic capability required and can as well manipulate files in an efficient manner. Also it is the most universally known computer language and is relatively simple to use.

Once FORTRAN was chosen, it was then necessary to restrict the scope of the language to be implemented in order to provide all the essential features while remaining within the size limitations of the ASQ/114 hardware. Implementation of the entire set of functions available in IBM FORTRAN IV is impractical. It is impractical in that at least 30% of the functions either are not compatible with the system hardware or are normally not utilized by the general purpose user. Such functions as NAMELIST, BLOCK DATA, EXTERNAL, EQUIVALENCE, and FIND are seldom used. Implied do loops, multiple subscripted arrays, and double precision arithmetic can significantly expand the language compiler's size. The sophisticated data formats associated with complex arithmetic operations are not required for the intended users of the proposed system. With these considerations and others in mind the BNF syntax of a FORTRAN compiler has been designed.

The major features of this version of FORTRAN are:

A. Integer arithmetic
B. Read and Write statements with variable format
C. IF Statement with boolean and comparison expressions
D. DO statement with variable arguments
E. GO TO statements
F. DATA statements
G. Procedure and Function calls
H. Single subscripted arrays

This version of FORTRAN has many limitations, some of specific design origin others from efforts to insure ease of implementation. There are as well some relaxations from the standard FORTRAN language. The specific characteristics of this FORTRAN language have been placed into four categories for analysis: severe limitations, restrictions, minor inconveniences, and standard relaxations.

A. SEVERE LIMITATIONS
    1. integer arithmetic only

B. RESTRICTIONS
    1. single subscripted arrays
    2. no COMMON, GLOBAL, EQUIVALENCE etc. statements
    3. few built in functions
    4. modified data statement format

C. MINOR INCONVENIENCES
    1. all labels are numbers and are followed by a colon
    2. COMMENT is a reserve word
    3. termination read (END -) statement eliminated
    4. only single precision arithmetic
    5. computed/arithmetic GO TO eliminated
    6. no continuation statements

D. RELAXATIONS
    1. program input is free field (no column dependence)
    2. variable names are alphanumeric

The effect of the severe limitation can be minimized by the efficient programmer who is aware of this deficiency. For the anticipated applications of the general purpose system within the aviation community, the restrictions are of little concern. The inconveniences and relaxations should produce nothing more than possible syntax errors in programming and thus do not affect the programmer's logic patterns in his programs. Thus the compiler produced from this syntax appears to well provide the general purpose user of the ASQ/114 system the necessary and desirable capabilities to effectively program and utilize the system.

The compiler's BNF syntax was designed for use in a table driven syntax directed compiler. The generation steps for this compiler resemble

the generation of the CP 901 assembler. The main difference between the two generations is that the total effort required to produce the inital simple compiler is significantly greater than that expended for the generation of the assembler.

The first step for the compiler generation is the development of a syntax analyzer. This analyzer performs a dual task. It firstly analyzes the BNF syntax to insure that the syntax is logically correct and no ambiguities exist. It then generates the parsing tables. The parsing tables are merely the decision tables of the compiler's grammar checker. These tables direct the checker to either hold the current program input symbol or to combine it with previous symbols. After the compiler scans the entire program input, producing an intermediate form of the program, then the second stage of the compiler is completed. All that remains to be done is to develop a set of code generators that emit the proper machine code for each input statement encountered during the scanning.

Thus there are three basic routines that are essential for the generation of a compiler. There must be a syntax analyzer that accepts a BNF syntax input and generates a set of parsing decision tables. Secondly there must be a syntax checker of the program input to verify that the input is syntactically in agreement with that of the BNF grammar. Finally for each BNF reduction in the checking routine, there must be a generator of the proper machine code for that reduction.

For the ASQ/114 system there is only one language to write the three compiler routines, the developed assembly language. It was for this initial generation of the FORTRAN compiler that the assembler was developed. Once these routines are written and the initial compiler is generated, then by the use of the bootstrapping technique newer versions of the

compiler can be developed. Eventually after many incremental additions and subsequent executions, the full range of the compiler as described by the syntax will be implemented.

IX. THE CONTROL OPERATING SYSTEM

A system control program is a set of computer routines that relieves the computer user of organizing and allocating all of the system resources (memory, registers, tapes, and I/O devices) for the tasks he wishes to execute. The control program automatically schedules and supervises the active task, controls the location and retrieval of data, and assures the most efficient processing of tasks through the system. A control program usually consists of several routines of which the Supervisor (monitor), Scheduler, and the I/O Controller are the major ones.

The Supervisor usually controls the actions of the other two, as well as other service routines, language processors and user programs. The service routines consist of housekeeping or utility routines and special application programs. Language processors may be either assemblers, interpreters, translators or compilers. At the present time the CP 901 general purpose system will not have any special application or utility routines nor any special housekeeping programs, other than those required by the Scheduler in manipulating the various tasks within the system. There will be an assembler, used primarily for system development, and a simple FORTRAN compiler.

The basic function of the Scheduler is the efficient manipulation of the system as directed by the general task description supplied by the user. Thus for simple task statements the Scheduler under the control of the Supervisor, will efficiently co-ordinate the necessary resources for the task execution. The Scheduler further interacts with

the I/O Controller to allocate the necessary I/O devices required by the task. The operation of these routines is totally unseen by the user.

All of these routines will constitute a simple Operating System. For the CP 901 the operating system will generally be small as compared to some of the operating systems of the large systems in use today, but none-the-less it should be as effective. This operating system will exploit the resources of the system enabling the user to solve a wide variety of tasks efficiently with the central processor.

Though difficult to estimate, it is believed that only about five hundred instructions will be required for the CP 901 operating system. These routines will permanently reside in memory beginning at address 620 octal, the current loading point of the bootstrap loader of the central processor.

The operating system will have access to one of the tape drives on which the assembler and compiler will reside. If the assigned task requires either of these routines, they must be read into a reserved section of memory and executed utilizing the designated input source and device. This source could either be one of the magnetic tapes or the system's keyboard terminal. If the source is on tape, then the operating system must allocate the second tape drive for input to the task. At this time then the compiler is read into core memory and its tape removed so that this tape drive may be utilized for the object code output from the compiler. Note that if the compiler source is from the keyboard terminal, the second tape drive may be used for the compiler output. Once the output from the compiler is collected on tape, the monitor (Supervisor) regains control of the central processor and awaits a new task. Perhaps the new task may be the execution of previously

compiled program. In this case the compiled program would be read from tape and the necessary I/O devices would be allocated as specified in the task statement. Execution of the program would then begin. Upon completion, the monitor would once again resume control of the processor.

In this manner all processing would be under the control of the operating system as directed by the task assignments of the user. These task assignments will be made using a standard and simple user oriented command language.

This operating system will relieve the user from many of the functions normally required of a general system user on a machine of this size and, in this manner, the CP 901 may be utilized to a much greater extent by the potential squadron users.

X. CONCLUSIONS AND RECOMMENDATIONS

Sufficient computer power is readily available in the CP 901 idle time to warrant efforts to utilize that time. There is also a rapidly expanding administrative and supply workload within the squadrons having this idle computer power to warrant these efforts. Providing the general purpose capability for the CP 901 that is installed in the P3C aircraft both allows utilization of the computer power and assists in the handling of the administrative workload.

Although the feasibility study was briefly conducted, it provided the only hardware system limitation, the lack of sufficient I/O equipment. The study's conclusion indicated that a general purpose operating system was feasible even with the I/O limitation.

The translator operates under the OS/360 operating system and requires 350K for execution. This routine has been tested only with controlled test packages. No known discrepancies exist in this design tool.

The CP 901 assembler presently executes under OS/360 or CP/CMS and produces CP 901 machine code for the various assembly language inputs. This routine has been tested using only controlled test packages. It is not totally coded but those uncoded assembly instruction were designated for future use in section VI. It is believed that the coded set of instructions represent an initial usable set in which reasonable programs could be coded.

The bootstrapping technique described in section VII and appendix B is brief but technically sound. Further description of this process is available in section 8.13 of Reference 3.

The design of the FORTRAN compiler appears reasonable and seems to adequately provide the required system capabilities. It is expected that significant efforts will still be required to produce a working CP 901 FORTRAN compiler.

Once the compiler, assembler, and the control operating system are operational, these software programs may then be combined to produce the end product, the ASQ/114 general purpose digital computer system.

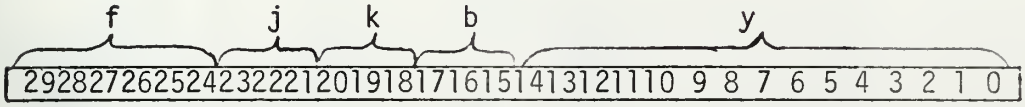
It is recognized that only the basic essentials have been investigated and reported. Many gaps exist and can only be filled with additional dedicated study of the system. It is further essential that this study be undertaken prior to any effort towards the installation of the general purpose ASQ/114 operating system.

It is recommended that followup efforts towards the implementation of the general purpose operating system be initiated either as a followup thesis study or a sponsored project. In this way implementation of a significantly beneficial package may be made available to all P3C squadrons.

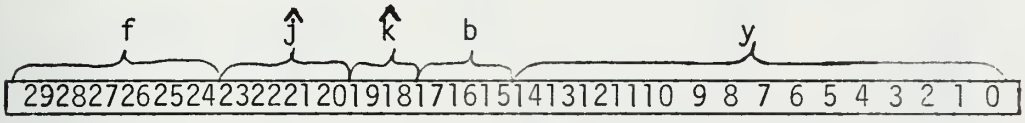
APPENDIX A

Instruction Word Format Types

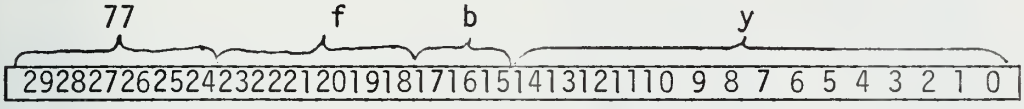
FORMAT I GENERAL INSTRUCTION



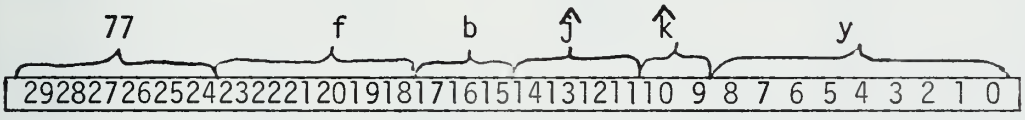
FORMAT I I/O INSTRUCTION WORD



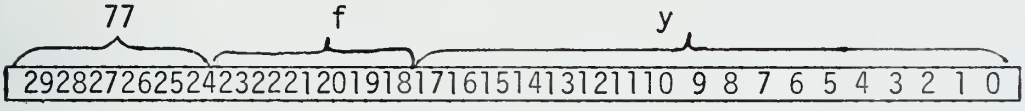
FORMAT II GENERAL INSTRUCTION WORD



FORMAT II SPECIAL INSTRUCTION WORD



FORMAT II DIRECT ADDRESSING INSTRUCTION WORD

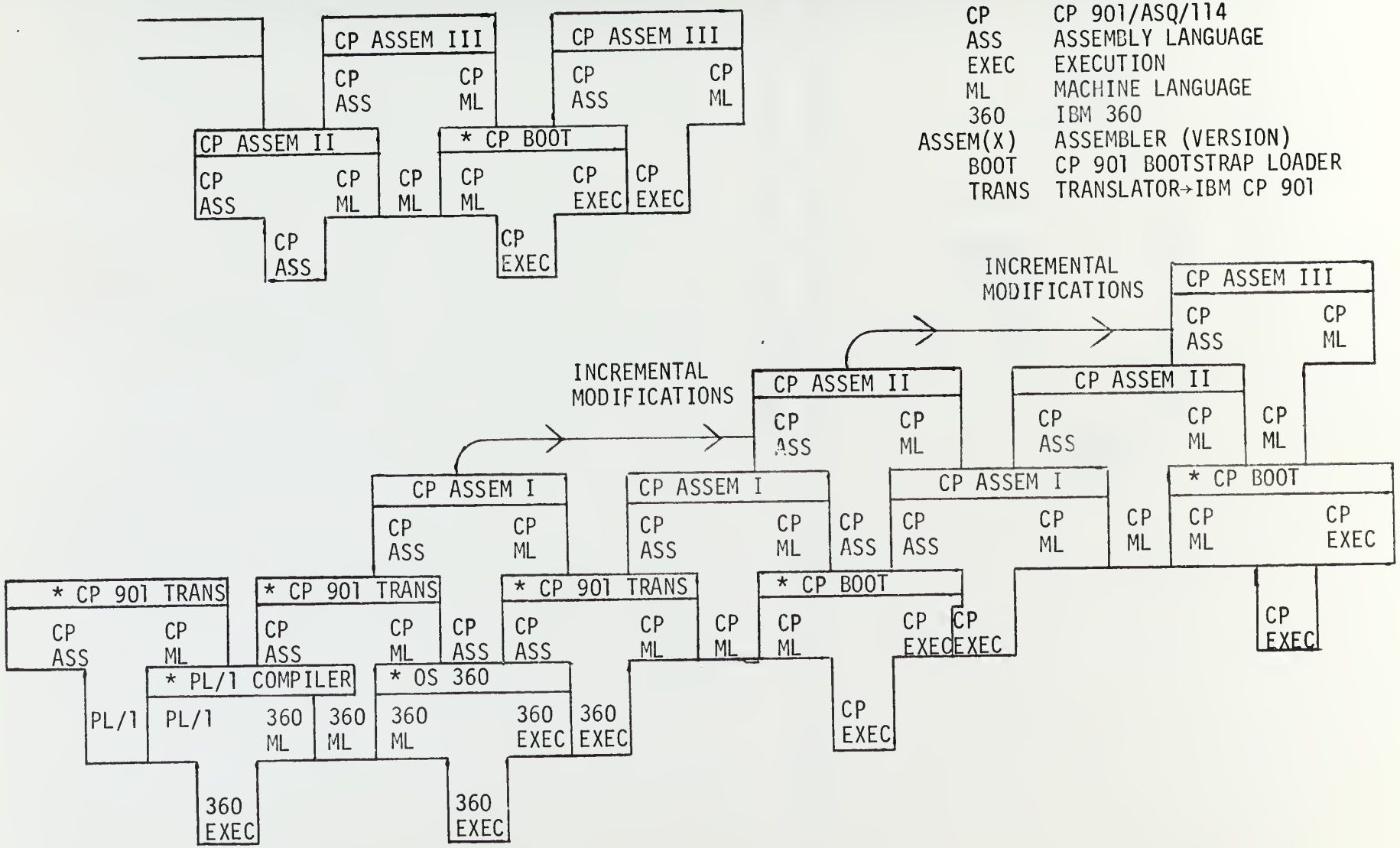


f --- opcode
y --- operand address
b --- b register designator

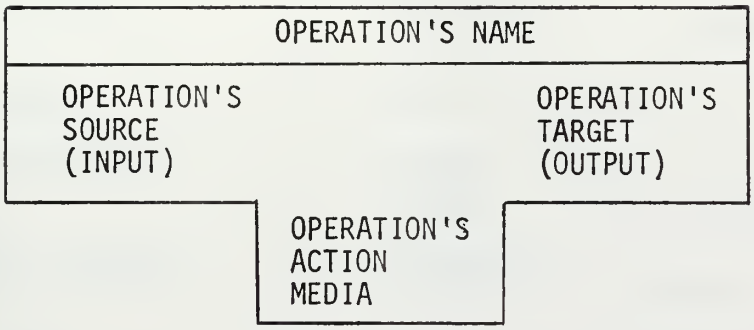
j --- j designator
k --- k designator
j^ --- j^ designator
k^ --- k^ designator

APPENDIX B

Bootstrapping Generation Scheme



Operation's Structure Description



Note that wherever the symbol (ML) appears, that task step could then be applied to the translator for test execution.

APPENDIX C

Assembler BNF Syntax

```
[ASSEMBLY]              ::=  [ASSEMBLY SET] END

[ASSEMBLY SET]          ::=  [STATEMENT]
                             [ASSEMBLY SET]  [STATEMENT]

[STATEMENT]             ::=  [BASIC STATEMENT]

[BASIC STATEMENT]       ::=  [NORMAL STATMENT]
                             [LABEL] : [NORMAL STATEMENT]

[NORMAL STATEMENT]      ::=  [OPERATION]
                             [OPERATION]  [OPERAND]

[OPERATION]             ::=  ADD|SUB|MUL|DIV|EXP|LES|EQL|
                             LEQ|NEQ|GEQ|GTR|NEG|NOT|AND|
                             BOR|ONE|TWO|LOD|STO|DEL|STD|
                             XIT|BFN|BFC|BBC|BRS|BSC|NOP|
                             PRO|RTN|GET|RET|RDV|WRV|WRS|
                             DMP|TAB|ROW|SCR|SAV|UNS|SUP|
                             LIT|EOF|DEL|DUP|XCH|048

[LABEL]                 ::=  540|541|542|543|544|545|546|547|
                             550|551|552|553|554|555|556|557|
                             560|561|562|563|564|565|566|567|
                             570|571|572|573|574|575|576|577

[OPERAND]               ::=  [NUMBER]

[NUMBER]                ::=  [DIGIT]
                             [NUMBER]  [DIGIT]

[DIGIT]                 ::=  0|1|2|3|4|5|6|7|8|9
```

APPENDIX D

FORTRAN Compiler BNF Syntax

```
[PROGRAM]              ::= [STRUCTURE]

[STRUCTURE]            ::= [STATEMENT LIST]  END

[STATEMENT LIST]       ::= [STATEMENT]
                           [STATEMENT LIST]  [STATEMENT]

[STATEMENT]            ::= [SIMPLE STATEMENT]
                           COMMENT  [ANYTHING]
                           [LABEL]  :  [SIMPLE STATEMENT]
                           [LABEL]  :  CONTINUE
                           [PROCEDURE DECLARATION]

[ANYTHING]             ::= [BETA SET]

[LABEL]                ::= [NUMBER]

[ALPHA SET]            ::= [CHARACTER SET]
                           [ALPHA SET]  [CHARACTER SET]

[BETA SET]             ::= [CHARACTER]
                           [BETA SET]  [CHARACTER]

[CHARACTER SET]        ::= [DIGIT]
                           [LETTER]

[CHARACTER]            ::= [DIGIT]
                           [LETTER]
                           [SYMBOL]

[NUMBER]               ::= [DIGIT]
                           [NUMBER]  [DIGIT]

[DIGIT]                ::= 0|1|2|3|4|5|6|7|8|9

[LETTER]               ::= A|B|C|D|E|F|G|H|I|etc.

[SYMBOL]               ::= +|=|*|¬|)|(|'|_|$|#|"|/|?|@|.|,|;|:

[SIMPLE STATEMENT]     ::= [DIMENSION STATEMENT]
                           [READ STATEMENT]
                           [WRITE STATEMENT]
                           [ASSIGNMENT STATEMENT]
                           [IF STATEMENT]
                           [DATA STATEMENT]
                           [FORMAT STATEMENT]
                           [DO STATEMENT]
                           [CONTROL STATEMENT]
```

```
[ASSIGNMENT STATEMENT    ::=  [VARIABLE] = [RIGHT PART]

[RIGHT PART]             ::=  [EXPRESSION]
                              [SUBSCRIPTED VARIABLE]

[EXPRESSION]             ::=  [ARITHMETIC EXPRESSION]
                              [VARIABLE]
                              [FUNCTION CALL]

[ARITHMETIC EXPRESSION]  ::=  [TERM]
                              [ARITHMETIC EXPRESSION] + [TERM]
                              [ARITHMETIC EXPRESSION] - [TERM]
                              - [TERM]

[TERM]                   ::=  [PRIMARY]
                              [TERM] * [PRIMARY]
                              [TERM] / [PRIMARY]
                              [TERM]  [INVOLUTION FACTOR]

[INVOLUTION FACTOR]      ::=  ** [PRIMARY]

[PRIMARY]                ::=  [VARIABLE]
                              [NUMBER]
                              (  [EXPRESSION]  )

[VARIABLE]               ::=  [IDENTIFIER]
                              [SUBSCRIPTED VARIABLE]

[IDENTIFIER]             ::=  [ALPHA SET]

[SUBSCRIPTED VARIABLE]   ::=  [SUBSCRIPTED HEAD]  [EXPRESSION]  )

[SUBSCRIPT HEAD]         :;=  [IDENTIFIER]  (

[DIMENSION STATEMENT]    ::=  DIMENSION [ARRAYS]

[ARRAY]                  ::=  [SUBSCRIPT HEAD]  [ARRAY LIST]

[ARRAYS]                 ::=  [ARRAY]
                              [ARRAYS]  ,  [ARRAY]

[ARRAY LIST]             ::=  [NUMBER]  )

[READ STATEMENT]         ::=  [READ HEAD]  )  [VARIABLE LIST]

[READ HEAD]              ::=  READ  (  [IO]

[IO]                     ::=  [NUMBER]  ,  [LABEL]

[VARIABLE LIST]          ::=  [VARIABLE]
                              [VARIABLE LIST]  ,  [VARIABLE]
```

```
[WRITE STATEMENT]       ::=  [WRITE HEAD]  )  [EXPRESSION LIST]
                             [WRITE HEAD]  )  [STRING LIST]

[WRITE HEAD]            ::=  WRITE  (  [IO]

[EXPRESSION LIST]       ::=  [EXPRESSION]
                             [EXPRESSION LIST]  ,  [EXPRESSION]

[STRING LIST]           ::=  [STRING]
                             [STRING LIST]  ,  [STRING]

[STRING]                ::=  '  [STRING BODY]  '

[STRING BODY]           ::=  [BETA STRING]

[IF STATEMENT]          ::=  [BOOLEAN PRIMARY]
                             [BOOLEAN EXPRESSION]  .AND.  [BOOLEAN PRIMARY]
                             [BOOLEAN EXPRESSION]  .OR.  [BOOLEAN PRIMARY]
                             .NOT.  [BOOLEAN PRIMARY]

[BOOLEAN PRIMARY]       ::=  [COMPARISON]
                             (  [BOOLEAN EXPRESSION]  )

[COMPARISON]            ::=  [EXPRESSION]  [RELATION]  [EXPRESSION]

[RELATION]              ::=  .EQL.|.NEQ.|.LES.|.LEQ.|.GTR.|.GEQ.

[THEN]                  ::=  [JUMP]
                             [ASSIGNMENT STATEMENT]
                             [PROCEDURE CALL]
                             STOP

[JUMP]                  ::=  [GO TO STATEMENT]

[DATA STATEMENT]        ::=  DATA [DATA MODE]

[DATA MODE]             ::=  [DATA DEF]
                             [DATA MODE]  ,  [DATA DEF]

[DATA DEF]              ::=  [ALPHA SET]  /  [INFO]

[INFO]                  ::=  [NUMBER]

[FORMAT STATEMENT]      ::=  [LABEL]  :  [FORMAT HEAD]

[FORMAT HEAD]           ::=  FORMAT  (  [FORMAT LIST]

[FORMAT LIST]           ::=  [FORMAT FIELD]  )

[FORMAT FIELD]          ::=  [FORMAT EXPRESSION]
                             [FORMAT FIELD]  ,  [FORMAT EXPRESSION]
```

```
[FORMAT EXPRESSION]      ::=  [ALPHA SET]
                              [MULTIPLE]  [ALPHA SET]

[MULTIPLE]               ::=  [NUMBER]

[GO TO STATEMENT]        ::=  GO TO [LABEL]
                              GOTO [LABEL]

[CONTROL STATEMENT]      ::=  STOP
                              RETURN

[DO STATEMENT]           ::=  DO [DO EXPRESSION]

[DO EXPRESSION]          ::=  [LABEL]  [DO FORMAT]

[DO FORMAT]              ::=  [VARIABLE]  =  [DO LIST]

[DO LIST]                ::=  [ARG]  ,  [ARG LIST]

[ARG LIST]               ::=  [ARG]
                              [ARG]  ,  [ARG]

[ARG]                    ::=  [VARIABLE]

[PROCEDURE DECLARATION]  ::=  [PROCEDURE HEAD]  [PROCEDURE BODY]

[PROCEDURE HEAD]         ::=  FUNCTION [PROCEDURE LIST]
                              PROCEDURE [PROCEDURE LIST]

[PROCEDURE LIST]         ::=  [PROCEDURE NAME]  (  [ARGUMENTS]

[PROCEDURE NAME]         ::=  [ALPHA SET]

[ARGUMENTS]              ::=  [ARGS]  )

[ARGS]                   ::=  [ARG]
                              [ARGS]  ,  [ARG]

[PROCEDURE BODY]         ::=  [STRUCTURE]

[FUNCTION CALL]          ::=  [PROCEDURE NAME]  (  [ARGUMENTS]

[PROCEUDRE CALL]         ::=  CALL [PROCEDURE LIST]
```

```
//RAY$SIM  JOB  (0369,0405NT,CS04),'DENNIS E. RAY',TIME=4
// EXEC PL1CLG,REGION.GO=350K
//PL1L.SYSIN DD *
 SIMUL: PROCEDURE OPTIONS (MAIN);                                        SIM00010
 DCL SYSPRINT FILE STREAM PRINT ENV (F(133));
 DCL SYSIN FILE STREAM ENV (F(80));
 DCL (A,A1,Q,Q1,U,Z,Y_BAR,INPUT) BIT (30);                               SIM00020
 DCL (F,FS) BIT (6);
 DCL AQ BIT (60);
 DCL (J,K,B,R) BIT (3);
 DCL (L,P_DEC,P_OCT,F_DEC,A_VAL,Q_VAL,YBAR_VAL) FIXED BINARY (15);       SIM00050
 DCL (B_DEC,J_DEC,K_DEC,R_DEC) FIXED BINARY (15);
 DCL J_HAT BIT (4);                                                      SIM00060
 DCL K_HAT BIT (2);                                                      SIM00070
 DCL (P,Y) BIT (15);                                                     SIM00080
 DCL Y_1 BIT (9);                                                        SIM00090
 DCL Y_2 BIT (16);                                                       SIM00100
 DCL INPUT4 CHAR (30) VAR;                                               SIM00110
 DCL INPUT1 CHAR (30);                                                   SIM00120
 DCL PARITY CHAR (4);                                                    SIM00130
 DCL BSTR BIT (1) INITIAL ('0'B);                                        SIM00140
 DCL MEM1 (0:32767) BIT (30);                                            SIM00150
 DCL MEM2 (0:32767) BIT (30);                                            SIM00160
 DCL FLAG BIT (1) INITIAL ('0'B);
 DCL (ASTR, CSTR) BIT (30) VAR;                                          SIM00170
 DCL VAR (77) LABEL;                                                     SIM00180
 DCL J_VAR (7) LABEL;                                                    SIM00190
 DCL B_REGIS (0:7) BIT (15);                                             TIM01320
 DCL (JUMP1,JUMP2,JUMP3,STOP5,STOP6,STOP7) BIT (1) INITIAL ('0'B);

     /*   PROCEDURE OCTAL RETURNS THE FIXED DEC REPRESENTATION
          OF THE THREE BIT INPUT IN OCTAL NOTATION.                  */

     /*   PROCEDURE REP_STR REPEATS THE FIRST PARAMETER SINGLE
         BIT THE NUMBER OF TIMES SPECIFIED BY THE SECOND PARAMETER.  */

     /*   PROCEDURE MEMORY IS USED TO ADDRESS THE UPPER AND
          LOWER HALF OF MEMORY BY THE SINGLE REFERENCE TO
          DUMMY MEMORY.                                              */

     /*   PROCEDURE VAL_THIRTY RETURNS A FIXED BINARY NUMBER
          REPRESENTED IN DECIMAL FORM IN OCTAL NOTATION.             */

     /*   PROCEDURE VAL_15 RETURNS A FIXED BINARY NUMBER IN OCTAL
          NOTATION FOR THE FIFTEEN BIT INPUT.                        */

     /*   PROCEDURE PAR_CHECK IS A PARITY CHECKER IN THAT IT
          COUNTS THE NUMBER OF 1 BITS IN A GIVEN BIT STRING
```

```
          OF THIRTY BITS AND RETURNS EITHER EVEN OR ODD.                */

     /*   PROCEDURE CIR_SFT  CIRCULARLY SHIFTS THE THIRTY BIT
          INPUT STRING THE NUMBER OF BITS SPECIFIED BY THE
          SECOND PARAMETER.                                             */

     /*   PROCEDURE REV_OCTAL RETURNS THE THREE BITS THAT
          REPRESENT THE INPUT NUMERIC CHARACTER                         */

     /*   PROCEDURE STORE STORES INTO THE DUMMY MEMORY THE FIRST
          PARAMETER AT THE ADDRESS GIVEN BY THE SECOND
          PARAMETER.                                                    */

     /*   PROCEDURES J_CHANGE READ_IT SPEC_J REP_STO REP_READ
          ALL MANIPULATE THE VARIOUS BIT PATTERNS AS DIRECTED
          BY THE MODIFIERS. EACH OF THESE PROCEDURES ACCOMPLISHES
          THE ACTUAL MODIFICATION OF ONE OF THE MODIFIERS.              */

DCL OCTAL ENTRY (BIT (3)) RETURNS (FIXED BINARY);                       SIM00210
DCL REP_STR ENTRY (BIT(1), FIXED BINARY (15)) ;                         SIM00220
DCL MEMORY ENTRY (FIXED BINARY (15)) RETURNS (BIT (30));                SIM00230
DCL VAL_THIRTY ENTRY (BIT (30)) RETURNS (FIXED BINARY (15));            SIM00240
DCL VAL_15 ENTRY (BIT(15)) RETURNS (FIXED BINARY (15));                 SIM00250
DCL PAR_CHECK ENTRY (BIT (30)) RETURNS (CHAR (4));                      SIM00260
DCL CIR_SFT ENTRY (BIT (30),FIXED BINARY (15)) RETURNS (BIT (30));      SIM00270
DCL REV_OCTAL ENTRY (CHAR (1)) RETURNS (BIT (3));                       SIM00280
DCL STORE ENTRY (BIT (30),BIT (15));                                    SIM00290
DCL J_CHANGE ENTRY;                                                     SIM00300
DCL READ_IT ENTRY;                                                      SIM00310
DCL SPEC_J ENTRY;                                                       TIM01900
DCL REP_STO ENTRY (BIT (30), BIT (15));                                 SIM00320
DCL REP_READ ENTRY;                                                     SIM00330
DCL (HOLD1,HOLD2,HOLD3,HOLD4,HOLD5) FIXED BINARY (15);                  SIM00340
DCL TEMP1 CHAR (30) VAR;                                                SIM00350
DCL TEMP2 CHAR (30) VAR;                                                SIM00360
                                                                        SIM00370
     START:
                                                                        TIM01230

     /*  INITIALIZE B REGISTERS TO 0  */

      DO L = 0 TO 7;                                                    TIM01350
      B_REGIS (L) = '000000000000000'B;                                 TIM01360
      END;                                                              TIM01370
                                                                        TIM01250
    /*      P HOLDS THE ADDRESS OF THE NEXT INSTRUCTION TO BE           TIM01400
            PROCESSED. P_DEC IS USED TO INCREMENT THE ADDRESS           TIM01410
            P AND IS INTERPRETED BY THE PROCEDURE (VAL_15)              TIM01420
```

```
          FOR CONVERSION TO OCTAL REPRESENTATION. U HOLDS THE          TIM01430
          CONTENTS OF THE ADDRESS SPECIFIED BY P AS THE                TIM01440
          INSTRUCTION IS EXECUTED ACCORDING TO THE VARIOUS             TIM01450
          FIELDS CONTAINED IN THE INSTRUCTION.                      */ TIM01460
    P = '0000001100100000'B;
    P_OCT = 620;
    P_DEC = 400;
       START2: PUT FILE (SYSPRINT) EDIT ('P = ',P) (SKIP,A,B(15));
               P_DEC = P;                                              SIM00580
         PUT FILE (SYSPRINT) EDIT ('P DEC IS ',P_DEC) (SKIP,A,F(6));
               P_OCT = VAL_15 (P);                                     SIM00600
         PUT FILE (SYSPRINT) EDIT ('P OCT = ',P_OCT) (SKIP,A,F(6));
START3: U = MEMORY (P_OCT);                                            SIM00620
      PUT FILE (SYSPRINT) EDIT ('U = ',U) (SKIP,A,B(30));
F = SUBSTR (U,1,6);                                                    SIM00640
B = SUBSTR (U,13,3);                                                   SIM00710
IF F = '111111' THEN GO TO DIRECT;                                     SIM00650
J = SUBSTR (U,7,3);                                                    SIM00670
J_HAT = SUBSTR (U,7,4);                                                SIM00680
K = SUBSTR (U,10,3);                                                   SIM00690
K_HAT = SUBSTR (U,11,2);                                               SIM00700
Y = SUBSTR (U,16,15);                                                  SIM00720
GO TO PROCESS;                                                         SIM00730
 DIRECT: FS = SUBSTR (U,7,6);
J_HAT = SUBSTR (U,16,4);                                               SIM00760
K_HAT = SUBSTR (U,20,2);                                               SIM00770
Y_1 = SUBSTR (U,22,9);                                                 SIM00780
Y_2 = SUBSTR (U,14,16);                                                SIM00790
PROCESS: F_DEC = 10 * OCTAL (SUBSTR (F,1,3)) + OCTAL (SUBSTR (F,4,3)); SIM00800
    PUT FILE (SYSPRINT) EDIT ('F DEC = ',F_DEC) (SKIP,A(8),F(8));
  PUT FILE (SYSPRINT) EDIT ('A REG IS ',A) (SKIP,A,B(30));
  PUT FILE (SYSPRINT) EDIT ('Q REG IS ',Q) (SKIP,A,B(30));
IF F_DEC = 0 THEN DO;                                                  SIM00820
          P_DEC = P_DEC + 1;                                           SIM00840
          P = BIT (P_DEC,15);                                          SIM00850
           P_DEC = VAL_15 (P);                                         SIM00860
              GO TO START2;                                            SIM00870
              END;                                                     SIM00880
   /*     THIS CREATES THE OPERAND (Y) FROM Y AND B OF THE             TIM01320
          INSTRUCTION WORD.                                         */ TIM01330
    Y = BIT (BINARY (Y,15,0) + BINARY (B_REGIS (OCTAL (B)),15,0),15);
GO TO VAR (F_DEC);                                                     SIM00890
                                                                       SIM00900
   /*    RIGHT SHIFT Q REGISTER                                       TIM00010
         HOLD1 IS THE SHIFT COUNTER                                    TIM00020
         ASTR CONTAINS THE PROPER NUMBER OF ZERO BITS     */          TIM00030
                                                                       TIM00410
VAR (1): CALL READ_IT;                                                 SIM00910
```

```
            HOLD1 = 10 * OCTAL (SUBSTR (Y_BAR,25,3)) +                      SIM00920
                        OCTAL (SUBSTR (Y_BAR,28,3));                        SIM00930
             CALL J_CHANGE ;                                                TIM00160
            CALL REP_STR (BSTR,HOLD1);                                      SIM00950
             Q = ASTR || SUBSTR (Q,1,30-HOLD1);                             TIM00170
            GO TO ENDING;                                                   SIM01000
                                                                            SIM01010
       /*   RIGHT SHIFT A REGISTER                                          TIM00050
            HOLD1 IS THE SHIFT COUNTER                                      TIM00060
            ASTR CONTAINS THE PROPER NUMBER OF ZERO BITS      */            TIM00070
                                                                            TIM00400
   VAR (2): CALL READ_IT;                                                   SIM01020
            CALL J_CHANGE;                                                  SIM01030
            HOLD1 = 10 * OCTAL (SUBSTR (Y_BAR,25,3)) +                      SIM01040
                         OCTAL (SUBSTR (Y_BAR,28,3));                       SIM01050
            CALL REP_STR (BSTR,HOLD1);                                      SIM01060
             A = ASTR || SUBSTR (A,1,30-HOLD1);                             TIM00210
            GO TO ENDING;                                                   SIM01120
                                                                            SIM01130
       /*   RIGHT SHIFT A & Q REGISTERS AS ONE REGISTER                     TIM00090
            HOLD1 IS THE SHIFT COUNTER                                      TIM00100
            TEMP1 HOLDS THE BITS TO BE SHIFTED INTO THE Q REGISTER          TIM00110
            ASTR CONTAINS THE PROPER NUMBER OF ZERO BITS      */            TIM00120
                                                                            TIM00380
   VAR (3): CALL READ_IT;                                                   SIM01140
            CALL J_CHANGE;                                                  SIM01150
          HOLD1 = 10 * OCTAL (SUBSTR (Y_BAR,25,3)) +                        SIM01160
                       OCTAL (SUBSTR (Y_BAR,28,3));                         SIM01170
            CALL REP_STR (BSTR,HOLD1);                                      SIM01200
             TEMP1 = SUBSTR (A,30-HOLD1,HOLD1);                             TIM00250
             A = ASTR || SUBSTR (A,1,30-HOLD1);                             TIM00260
             Q = TEMP1 || SUBSTR (Q,1,30-HOLD1);                            TIM00270
            GO TO ENDING;                                                   SIM01260
                                                                            SIM01270
       /*    THIS INSTRUCTION COMPARES THE VALUE OF Y_BAR WITH THE          TIM00340
             VALUE OF THE A OR Q REGISTER. THE BRANCH CONDITION DESIGNATOR  TIM00350
             J_DEC DETERMINES WHETHER THE NEXT INSTRUCTION IS TO BE         TIM00360
             SKIPPED OR NOT IN COMPARISON TO THE VARIOUS VALUES.  */        TIM00370
                                                                            TIM00330
   VAR (4): CALL READ_IT;                                                   SIM01280
            J_DEC = OCTAL (J);                                              SIM01290
            YBAR_VAL = VAL_THIRTY (Y_BAR);                                  SIM01310
             A_VAL = VAL_THIRTY (A);                                        TIM00320
             Q_VAL = VAL_THIRTY (Q);                                        TIM00310
            IF J_DEC = 0 THEN GO TO ENDING;                                 SIM01370
            IF J_DEC = 1 THEN GO TO SKIP;                                   SIM01380
             GO TO J_VAR (J_DEC);                                           SIM01390
     J_VAR (2): IF YBAR_VAL ¬> Q_VAL THEN GO TO SKIP;                       SIM01400
```

```
                GO TO ENDING;                                          SIM01410
      J_VAR (3): IF YBAR_VAL > Q_VAL THEN GO TO SKIP;                  SIM01420
                GO TO ENDING;                                          SIM01430
      J_VAR (4): IF (Q_VAL ¬< YBAR_VAL & YBAR_VAL > A_VAL)             SIM01440
                                THEN GO TO SKIP;                       SIM01450
                GO TO ENDING;                                          SIM01460
      J_VAR (5): IF (YBAR_VAL > Q_VAL | YBAR_VAL ¬> A_VAL)             SIM01470
                                THEN GO TO SKIP;                       SIM01480
                GO TO ENDING;                                          SIM01490
      J_VAR (6): IF YBAR_VAL ¬> A_VAL THEN GO TO SKIP;                 SIM01500
                GO TO ENDING;                                          SIM01510
      J_VAR (7): IF YBAR_VAL > A_VAL THEN GO TO SKIP;                  SIM01520
                GO TO ENDING;                                          SIM01530
       SKIP: P_DEC = P_DEC + 1;                                        SIM01540
             P = BIT (P_DEC,15);                                       SIM01550
             GO TO ENDING;                                             SIM01560
                                                                       SIM01570
   /*      THIS INSTRUCTION SHIFTS REGISTER Q CIRCULARLY              TIM00470
           TO THE LEFT  (HOLD1) POSITIONS.THE                         TIM00480
           PROCEDURE (CIR_SFT) ACCOMPLISHES THE SHIFT.          */    TIM00490
                                                                      TIM00300
VAR (5): CALL READ_IT;                                                SIM01580
         CALL J_CHANGE;                                               SIM01590
         HOLD1 = 10 * OCTAL (SUBSTR (Y_BAR,25,3)) +                   TIM00430
                      OCTAL (SUBSTR (Y_BAR,28,3));                    TIM00440
         Q = CIR_SFT (Q,HOLD1);                                       TIM00450
         GO TO ENDING;                                                SIM01700
                                                                      SIM01710
   /*      THIS INSTRUCTION SHIFTS REGISTER A CIRCULARLY              TIM00580
           TO THE LEFT (HOLD1) POSITIONS. THE PROCEDURE               TIM00590
           (CIR_SFT) ACCOMPLISHES THE SHIFT.                  */     TIM00600
                                                                      TIM00570
VAR (6): CALL READ_IT;                                                SIM01720
         CALL J_CHANGE;                                               SIM01730
         HOLD1 = 10 * OCTAL (SUBSTR (Y_BAR,25,3)) +                   TIM00540
                      OCTAL (SUBSTR (Y_BAR,28,3));                    TIM00550
         A = CIR_SFT (A,HOLD1);                                       TIM00560
         GO TO ENDING;                                                SIM01810
                                                                      SIM01820
   /*      THIS INSTRUCTION SHIFTS THE A & Q REGISTERS TO THE         TIM00720
           CIRCULARLY AS ONE SIXTY BIT REGISTER. THE UPPER            TIM00730
           BITS OF A (TEMP1) ARE SHIFTED INTO THE LOWER BITS          TIM00740
           BITS OF Q.                                         */     TIM00750
                                                                      TIM00700
VAR (7): CALL READ_IT;                                                SIM01830
         CALL J_CHANGE;                                               SIM01840
         HOLD1 = 10 * OCTAL (SUBSTR (Y_BAR,25,3)) +                   TIM00640
                      OCTAL (SUBSTR (Y_BAR,28,3));                    TIM00650
```

```
            TEMP1 = SUBSTR (A,1,HOLD1);                                    TIM00670
            A = SUBSTR (A,HOLD1 + 1) || SUBSTR (Q,1,HOLD1);                TIM00680
            Q = SUBSTR (Q,HOLD1 + 1) || TEMP1;                             TIM00690
            GO TO ENDING;                                                  SIM01880
                                                                           SIM01890
      /*      THIS INSTRUCTION ENTERS THE VALUE Y_BAR INTO THE             TIM00780
              Q REGISTER.                                          */      TIM00790
                                                                           TIM00760
VAR (10):   CALL READ_IT;                                                  SIM01900
            CALL J_CHANGE;                                                 SIM01910
            Q = Y_BAR;                                                     SIM01920
            GO TO ENDING;                                                  SIM01930
                                                                           SIM01940
      /*      THIS INSTRUCTION ENTERS THE VALUE Y_BAR INTO THE             TIM00830
              A REGISTER.                                          */      TIM00840
                                                                           TIM00820
VAR (11):   CALL READ_IT;                                                  SIM01950
            CALL J_CHANGE;                                                 SIM01960
            A = Y_BAR;                                                     SIM01970
            GO TO ENDING;                                                  SIM01980
                                                                           SIM01990
      /*      THIS INSTRUCTION TRANSMITS THE VALUE OF THE LOWER            TIM00900
              FIFTEEN BITS OF THE OPERAND ADDRESS (Y_BAR) INTO             TIM00910
              THE B REGISTER DESIGNATED BY THE VALUE OF J (J_DEC).  */     TIM00920
VAR (12):   CALL READ_IT;                                                  SIM02000
            J_DEC = OCTAL (J);                                             TIM00870
            B_REGIS (J_DEC) = SUBSTR (Y_BAR,16,15);                        TIM00880
            GO TO ENDING;                                                  SIM02020
                                                                           SIM02030
    /*      THIS INSTRUCTION ESTABLISHES ONE WORD OF OUTPUT
            (MEMORY (Y)) AND DESIGNATES THE OUTPUT CHANNEL (J HAT).
            THE ADDRESS Y IS STORED IN LOCATION 140 PLUS J HAT.
            FOR THIS IMPLEMENTATION ONE WORD IS PLACED INTO FILE
            SYSPRINT WITH APPROPRIATE IDENTIFICATION. MONITOR
            AND FORCE FUNCTIONS AS DESIGNATED BY K HAT ARE
            IGNORED FOR IMPLEMENTATION.                                */

VAR (13):PUT FILE (SYSPRINT) EDIT ('OUTPUT BUFFER CREATION') (SKIP,A);
      MEM1 (140 + (OCTAL ('00'B || SUBSTR (J_HAT,1,1)) * 10 +
                   OCTAL (SUBSTR (J_HAT,2,3))))  = MEMORY (VAL_15 (Y));
      PUT FILE (SYSPRINT) EDIT
    ('OUTPUT MEMORY OF Y ', MEMORY (VAL_15(Y)), ' EQUALS ',
            VAL_THIRTY (MEMORY (VAL_15 (Y)))) (SKIP,A,B,A,F(8));
            GO TO ENDING;                                                  SIM02050
                                                                           SIM02060
      /*      THIS INSTRUCTION STORES THE CONTENTS OF THE Q                TIM01040
              REGISTER AT THE STORAGE ADDRESS Y AS MODIFIED                TIM01050
              BY THE DESIGNATOR K. FOR K = 0 COMPLEMENT REGISTER           TIM01060
```

```
            Q OR FOR K = 4 SET THE A REGISTER = Q REGISTER.                TIM01070
            THE COMPLEMENTING IS ACCOMPLISHED BY A CALL TO THE             TIM01080
            PL/1 PROCEDURE (¬). K_DEC IS THE VALUE OF THE                  TIM01090
            DESIGNATOR K. THE STORING IS ACCOMPLISHED BY                   TIM01100
            THE CALL TO PROCEDURE (STORE).                          */     TIM01110
                                                                           TIM01030
 VAR (14): K_DEC = OCTAL (K);                                              TIM00980
           IF K_DEC = 0 THEN Q = ¬ (Q);                                    TIM00990
             ELSE IF K_DEC = 4 THEN A = Q;                                 TIM01000
             ELSE CALL STORE (Q,Y);                                        TIM01010
           CALL J_CHANGE;                                                  TIM01020
           GO TO ENDING;                                                   SIM02130
                                                                           SIM02140
    /*     THIS INSTRUCTION STORE THE CONTENTS OF THE A                   TIM01200
           REGISTER AT THE STORAGE ADDRESS Y AS MODIFIED                  TIM01210
           BY THE DESIGNATOR K. FOR K = 0 SET THE Q REGISTER              TIM01220
           = TO THE A REGISTER. FOR K = 4 COMPLEMENT A.                   TIM01230
           THE COMPLEMENTING IS ACCOMPLISHED BY A CALL TO                 TIM01240
           THE PL/1 PROCEDURE (¬). K_DEC IS THE VALUE OF THE              TIM01250
           DESIGNATOR K. THE STORING IS ACCOMPLISHED BY THE               TIM01260
           CALL TO THE PROCEDURE (STORE).                          */     TIM01270
                                                                           TIM01190
 VAR (15): K_DEC = OCTAL (K);                                              TIM01140
           IF K_DEC = 0 THEN Q = A;                                        TIM01150
             ELSE IF K_DEC = 4 THEN A = ¬ (A);                             TIM01160
             ELSE CALL STORE (A,Y);                                        TIM01170
           CALL J_CHANGE;                                                  TIM01180
           GO TO ENDING;                                                   SIM02210
                                                                           SIM02220
    /*     THIS INSTRUCTION STORES A THIRTY BIT WORD WHOSE                TIM01360
           LOWER FIFTEEN BITS ARE ZEROS AND WHOSE UPPER                   TIM01370
           FIFTEEN BITS ARE THE CONTENTS OF THE B REGISTER                TIM01380
           DESIGNATED BY THE VALUE OF THE J DESIGNATOR.                   TIM01390
           THE STORING IS ACCOMPLISHED BY A CALL TO THE                   TIM01400
           PROCEDURE (STORE) WITH THE STORING                             TIM01410
           ADDRESS Y AS ONE OF THE PARAMETERS.                     */     TIM01420
                                                                           TIM01350
 VAR (16): CALL REP_STR ('0'B,15);                                         SIM02230
           CALL STORE (ASTR || B_REGIS (OCTAL (J)),Y);                     SIM02240
           GO TO ENDING;                                                   SIM02250
                                                                           SIM02260
    /*  THIS INSTRUCTION UTILIZES THE K HAT DESIGNATOR AS
FOLLOWS: K HAT = 3 TRANSFER INTERRUPT WORD FROM ADDRESS 520 +
          J TO MEMORY LOCATION Y,  FO K HAT = 2 INPUT ON C CHANNEL
          J HAT TO MAMORY LOCATION Y.  FOR K HAT = 1 OR 0 THE
NORMAL EXTERNAL FUNCTION &UFFER IS CHECKED FOR ACTIVITY. IF
          ACTIVE Y DETERMINES THE JUMP ADDRESS.  K HAT = 0,1,3 ARE
          NOT IMPLEMENTED AS THEY ARE SPECIAL PURPOSE.              */
```

```
VAR (17):PUT FILE (SYSPRINT) EDIT ('STORE CN OT TEST EFB') (SKIP,A);
    K_DEC = OCTAL ('0'B || K_KAT);
    IF K_DEC = 0 | K_DEC = 1 THEN DO;
    PUT FILE (SYSPRINT) EDIT ('INPUT WITH K HAT = 0 OR 1  NOT USED')
          (SKIP,A);
        GO TO ENDING;
       END;
    IF K_DEC = 3 THEN
    PUT FILE (SYSPRINT) EDIT ('SIMULATE INTERRUPT WORD TRANSFER ')
                    (SKIP,A);
    ELSE DO;
    PUT FILE (SYSPRINT) EDIT ('INPUT USING CHANNEL ',J_HAT)
                (SKIP,A,B);
     HOLD4 = VAL_15 (Y);
         GET FILE (SYSIN) EDIT (INPUT1) (A(30));
    IF HOLD4 > 32767 THEN
         MEM2 (HOLD4 - 32767) = BIT (INPUY1,30);
    ELSE MEM1 (HOLD4) = BIT (INPUT1,30);
    END;
          GO TO ENDING;                                                  SIM02290
                                                                         SIM02300
   /*     THIS INSTRUCTION ADDS TO THE A REGISTER THE CONTENTS           TIM01460
          OF THE OPERAND Y_BAR NOTE THE USE OF THE                       TIM01470
          EXPLICIT CALLS TO THE BINARY AND BIT CONVERTER OF PL/1.   */  TIM01480
                                                                         TIM01450
VAR (20): CALL READ_IT;                                                  SIM02310
          CALL J_CHANGE;                                                 SIM02320
          A = BIT (BINARY (Y_BAR,31,0) + BINARY (A,31,0),30);            SIM02330
          GO TO ENDING;                                                  SIM02340
                                                                         SIM02350
   /*     THIS INSTRUCTION SUBTRACTS FROM THE A REGISTER                 TIM01520
          THE CONTENTS OF THE OPERAND Y_BAR. NOTE THE USE OF             TIM01530
          THE EXPLICIT CALLS TO THE BINARY AND BIT CONVERTERS            TIM01540
          OF PL/1.                                                  */   TIM01550
                                                                         TIM01560
VAR (21): CALL READ_IT;                                                  SIM02360
          A = BIT (BINARY (A,31,0) - BINARY (Y_BAR,31,0),30);            SIM02370
          CALL J_CHANGE;                                                 SIM02380
          GO TO ENDING;                                                  SIM02390
                                                                         SIM02400
    /*  THIS INSTRUCTION MULTIPLIES THE CONTENTS OF THE Q REGISTER
         WITH THE K MODIFIED Y OPERAND AND LEAVES THE RESULT
         IN ONE SIXTY BIT REGISTER AQ.  A = UPPER THIRTY BITS
         Q = LOWER THIRTY BITS                                      */

VAR (22): CALL READ_IT;                                                  SIM02410
    AQ = BIT (VAL_THIRTY (Q) * VAL_THIRTY (Y_BAR),60);
```

```
      A = SUBSTR (AQ,1,30);
      Q = SUBSTR (AQ,31,30);
       CALL J_CHANGE;                                              SIM02420
       GO TO ENDING;                                               SIM02430
                                                                   SIM02440
  /*    THIS INSTRUCTION DIVIDES THE DOUBLE LENGTH REGISTER
        AQ BY THE K MODIFIED Y OPERAND.  THE QUOTIENT IS LEFT
        IN THE Q REGISTER, THE REMAINDER IN THE A REGISTER.
        FOR K = 7 TAKE THE SQUARE OF AQ LEAVING THE ROOT IN
        Q REGISTER AND REMAINDER IN THE A REGISTER.            */

VAR (23): CALL READ_IT;                                            SIM02450
    AQ = A || Q;
    K_DEC = OCTAL (K);
    IF K_DEC = 7 THEN DO;
      PUT FILE (SYSPRINT) EDIT ('TAKING SQRT')  (SKIP,A);
    END;
    ELSE DO;
      PUT FILE (SYSPRINT) EDIT ('DIVIDING AQ')  (SKIP,A);
     Q = BIT (BINARY (AQ,31,0) / VAL_THIRTY (Y_BAR),30);
    END;
       GO TO ENDING;                                               SIM02460
                                                                   SIM02470
  /*    THIS INSTRUCTION ADDS THE VALUE OF Y TO THE A              TIM01610
        REGISTER AND THEN STORES THE CONTENTS OF THE A             TIM01620
        REGISTER AT THE STORAGE ADDRESS DESIGNATED BY              TIM01630
        Y AND THE K DESIGNATOR. THE STORING IS ACCOMPLISHED        TIM01640
        BY THE CALL TO THE FUNCTION (REP_STO).                */   TIM01650
                                                                   TIM01660
VAR (24): A = BIT (BINARY (Y,31,0) + BINARY (A,31,0),30);          TIM01590
       CALL J_CHANGE;                                              SIM02490
      R_DEC = OCTAL (R);
      IF R_DEC > 4 & FLAG = '1'B THEN
       Y = BIT (BINARY (Y,15,0) + BINARY (B_REGIS (6),15,0),15);
       CALL REP_STO (A,Y);                                         SIM02500
       GO TO ENDING;                                               SIM02510
                                                                   SIM02520
  /*    THIS INSTRUCTION SUBTRACTS THE VALUE OF Y FROM             TIM01720
        THE (A) REGISTER AND THEN STORES THE VALUE OF              TIM01730
        THE (A) REGISTER AT THE STORAGE ADDRESS DESIGNATED         TIM01740
        BY Y AND THE K DESIGNATOR. THE STORING IS                  TIM01750
        ACCOMPLISHED BY THE CALL TO THE PROCEDURE (REP_STO).  */   TIM01760
                                                                   TIM01770
VAR (25): A = BIT (BINARY (A,31,0) - BINARY (Y,31,0),30);          TIM01700
      R_DEC = OCTAL (R);
      IF R_DEC > 4 & FLAG = '1'B THEN
       Y = BIT (BINARY (Y,15,0) + BINARY (B_REGIS (6),15,0),15);
        CALL REP_STO (A,Y);                                        SIM02550
```

```
            CALL J_CHANGE;                                                  SIM02560
            GO TO ENDING;                                                   SIM02570
                                                                            SIM02580
   /*     THIS INSTRUCTION ADDS THE VALUE OF THE OPERAND                   TIM01810
          Y_BAR TO THE CONTENTS OF THE Q REGISTER.                  */      TIM01820
                                                                            TIM01830
VAR (26): CALL READ_IT;                                                     SIM02590
          Q = BIT (BINARY (Q,15,0) + BINARY (Y_BAR,15,0),30);               SIM02600
          CALL J_CHANGE;                                                    SIM02610
          GO TO ENDING;                                                     SIM02620
                                                                            SIM02630
  /*  THIS INSTRUCTION SUBTRACTS THE CONTENTS OF THE K MODIFIED
        OPERAND Y FROM THE CONTENTS OF THE Q REGISTER.              */

VAR (27): CALL READ_IT;                                                     SIM02640
          Q = BIT (BINARY (Q,15,0) - BINARY (Y_BAR,15,0),30);               TIM01860
       CALL SPEC_J;                                                         SIM02660
          GO TO ENDING;                                                     SIM02670
                                                                            TIM02090
   /*     THIS INSTRUCTION ADDS THE VALUE OF THE OPERAND Y                 TIM02100
          TO THE CONTENTS OF THE Q REGISTER AND PLACES THE                  TIM02110
          RESULT INTO THE A REGISTER.                               */      TIM02120
                                                                            SIM02680
VAR (30): A = BIT (BINARY (Q,15,0) + BINARY (Y,15,0),30);                   SIM02690
          CALL J_CHANGE;                                                    SIM02700
          CALL READ_IT;                                                     SIM02710
          GO TO ENDING;                                                     SIM02720
                                                                            TIM02150
   /*     THIS INSTRUCTION SUBTRACTS THE VALUE OF THE                      TIM02160
          OPERAND Y FROM THE CONTENTS OF THE Q REGISTER AND                 TIM02170
          THEN COMPLEMENTS THE RESULT AND PLACES IT INTO THE                TIM02180
          A REGISTER.                                               */      TIM02190
                                                                            SIM02730
VAR (31): A = BIT (BINARY (Q,15,0) - BINARY (Y,15,0),30);                   SIM02740
          CALL J_CHANGE;                                                    SIM02750
          A = ¬ (A);                                                        SIM02760
          CALL READ_IT;                                                     SIM02770
          GO TO ENDING;                                                     SIM02780
                                                                            TIM02220
   /*     THIS INSTRUCTION ADDS THE CONTENTS OF BOTH THE A                 TIM02230
          AND THE Q REGISTERS PLACES THE RESULT INTO THE A                 TIM02240
          REGISTER. IT THEN STORES THE CONTENTS OF THE A                   TIM02250
          REGISTER AT THE STORAGE ADDRESS OF THE OPERAND Y                 TIM02260
          AS DESIGNATED BY THE DESIGNATOR K. THE STORING                   TIM02270
          IS ACCOMPLISHED BY THE CALL TO THE PROCEDURE                      TIM02280
          (STORE).                                                  */      TIM02290
                                                                            SIM02790
VAR (32): A = BIT (BINARY (A,15,0) + BINARY (Q,15,0),30);
```

```
            CALL J_CHANGE;                                                  SIM02800
            CALL STORE (A,Y);                                               SIM02810
            GO TO ENDING;                                                   SIM02820
                                                                            SIM02830
   /*       THIS INSTRUCTION SUBTRACTS THE CONTENTS OF THE                  TIM02320
            Q REGISTER FROM THE A REGISTER AND STORES THE RESULT            TIM02330
            INTO THE ADDRESS SPECIFIED BY THE OPERAND Y AND THE             TIM02340
            DESIGNATOR K. THE STORING IS ACCOMPLISHED BY THE CALL           TIM02350
            TO THE PROCEDURE (STORE).                                 */    TIM02360
                                                                            TIM02370
VAR (33):   A = BIT (BINARY (A,15,0) - BINARY (Q,15,0),30);                 SIM02840
            CALL J_CHANGE;                                                  SIM02850
            CALL STORE (A,Y);                                               SIM02860
            GO TO ENDING;                                                   SIM02870
                                                                            TIM00040
   /*       THIS INSTRUCTION ADDS THE CONTENTS OF Y TO THE                  TIM00050
            CONTENTS OF THE Q REGISTER AND PLACES THE RESULT                TIM00060
            IN THE A REGISTER. THE A REGISTER CONTENTS ARE THEN             TIM00070
            STORED AT STORAGE ADDRESS Y. THE STORING IS                     TIM00080
            ACCOMPLISHED BY THE CALL TO THE PROCEDURE (REP_STO).            TIM00090
            (REP_STO) UTILIZES THE K DESIGNATOR AS A MODIFIER.    */       TIM00100
                                                                            SIM02880
VAR (34):   A = BIT (BINARY (Q,15,0) + BINARY (Y,15,0),30);                 TIM00030
          R_DEC = OCTAL (R);
          IF R_DEC > 4 & FLAG = '1'B THEN
            Y = BIT (BINARY (Y,15,0) + BINARY (B_REGIS (6),15,0),15);
            CALL REP_STO (A,Y);                                             SIM02910
            CALL J_CHANGE;                                                  SIM02920
            GO TO ENDING;                                                   SIM02930
                                                                            SIM02940
   /*       THIS INSTRUCTION SUBSTRACTS THE CONTENTS OF THE                 TIM00160
            OPERAND Y FROM THE CONTENTS OF THE Q REGISTER                   TIM00170
            AND PLACES THE RESULTS INTO THE A REGISTER. THE                 TIM00180
            CONTENTS OF THE A REGISTER IS STORED AT THE ADDRESS             TIM00190
            Y. THE STORING IS ACCOMPLISHED BY THE CALL TO THE               TIM00200
            PROCEDURE (REP_STO) WHICH MAKES USE OF THE K                    TIM00210
            MODIFIER. NOTE THAT REGISTER A IS COMPLIMENTED PRIOR            TIM00220
            TO THE CALL TO (REP_STO).                                 */    TIM00230
                                                                            TIM00240
VAR (35):   A = BIT (BINARY (Q,15,0) - BINARY (Y,15,0),30);                 TIM00140
            A = ¬ (A);                                                      SIM02970
          R_DEC = OCTAL (R);
          IF R_DEC > 4 & FLAG = '1'B THEN
            Y = BIT (BINARY (Y,15,0) + BINARY (B_REGIS (6),15,0),15);
            CALL REP_STO (A,Y);                                             SIM02980
            CALL J_CHANGE;                                                  SIM02990
            GO TO ENDING;                                                   SIM03000
                                                                            SIM03010
```

```
   /*     THIS INSTRUCTION ADDS ONE TO THE CONTENTS OF THE                TIM00290
          OPERAND Y AND PLACES THE RESULT INTO THE A REGISTER.            TIM00300
          THE CONTENTS OF THE A REGISTER ARE THEN STORED                  TIM00310
          AT STORAGE ADDRESS Y USING THE CALL TO PROCEDURE                TIM00320
          (REP_STO) AS MODIFIED BY THE K DESIGNATOR.            */        TIM00330
                                                                          TIM00340
VAR (36): A = BIT (BINARY (Y,15,0) + 1 ,30);
          CALL J_CHANGE;                                                  SIM03030
          R_DEC = OCTAL (R);
          IF R_DEC > 4 & FLAG = '1'B THEN
          Y = BIT (BINARY (Y,15,0) + BINARY (B_REGIS (6),15,0),15);
          CALL REP_STO (A,Y);                                             SIM03040
          GO TO ENDING;                                                   SIM03050
                                                                          SIM03060
   /*     THIS INSTRUCTION SUBTRACTS THE VALUE OF THE                     TIM00390
          OPERAND Y FROM THE ONE AND PLACES THE                           TIM00400
          RESULT INTO THE A REGISTER. THE A REGISTER IS                   TIM00410
          THEN COMPLIMENTED AND STORED AT THE STORAGE ADDRESS             TIM00420
          Y AS MODIFIED BY THE K DESIGNATOR.                    */        TIM00430
                                                                          TIM00440
VAR (37): A = BIT (1 - BINARY (Y,15,0),30);                               TIM00370
          A = ¬ (A);                                                      SIM03080
          CALL J_CHANGE;                                                  SIM03090
          R_DEC = OCTAL (R);
          IF R_DEC > 4 & FLAG = '1'B THEN
          Y = BIT (BINARY (Y,15,0) + BINARY (B_REGIS (6),15,0),15);
          CALL REP_STO (A,Y);                                             SIM03100
          GO TO ENDING;                                                   SIM03110
                                                                          SIM03120
   /*     THIS INSTRUCTION ENTERS THE BIT BY BIT PRODUCT OF               TIM00500
          Y_BAR AND THE CONTENTS OF THE Q REGISTER INTO THE               TIM00510
          A REGISTER. THE PARITY OF THE A REGISTER IS CHECKED             TIM00520
          BY THE CALL TO THE PROCEDURE (PAR_CHECK). IF THE PARITY         TIM00530
          IS EVEN AND THE VALUE OF THE J DESIGNATOR (J_DEC)               TIM00540
          IS 2 THEN SKIP THE NEXT INSTRUCTION OR IF THE                   TIM00550
          PARITY IS ODD AND THE VALUE OF THE J DESIGNATOR IS              TIM00560
          3 THEN SKIP THE NEXT INSTRUCTION. NOTE THE                      TIM00570
          USE OF THE PL/1 FUNCTION (BOOL).                      */        TIM00580
                                                                          TIM00590
VAR (40): CALL READ_IT;                                                   SIM03130
          A = BOOL (Y_BAR,Q,'0001');                                      SIM03140
          PARITY = PAR_CHECK (A);                                         SIM03150
          J_DEC = OCTAL (J);                                              TIM00460
          IF J_DEC = 2 & PARITY = 'EVEN' THEN GO TO SKIP;                 TIM00470
          IF J_DEC = 3 & PARITY = 'ODD ' THEN GO TO SKIP;                 TIM00480
          GO TO ENDING;                                                   SIM03180
                                                                          SIM03190
   /*     THIS INSTRUCTION ADDS TO THE CONTENTS OF THE A                  TIM00620
```

```
          REGISTER THE BIT BY BIT PRODUCT OF THE OPERAND                    TIM00630
          Y_BAR AND THE CONTENTS OF THE Q REGISTER. NOTE THE                TIM00640
          USE OF THE PL/1 FUNCTIONS (BIT,BOOL,BINARY).              */      TIM00650
                                                                            TIM00660
VAR (41): CALL READ_IT;                                                     SIM03200
          A = BIT (BINARY (A,15,0) +                                        SIM03210
                   BINARY (BOOL (Y_BAR,Q,'0001'),15,0),30);                 SIM03220
          CALL J_CHANGE;                                                    SIM03230
          GO TO ENDING;                                                     SIM03240
                                                                            SIM03250
   /*     THIS INSTRUCTION SUBTRACTS FROM THE CONTENTS OF                   TIM00690
          THE A REGISTER THE BIT BY BIT PRODUCT OF THE                      TIM00700
          OPERAND Y_BAR AND THE CONTENTS OF THE Q REGISTER.                 TIM00710
          NOTE THE USE OF THE PL/1 FUNCTIONS (BIT,BOOL,BINARY).      */TIM00720
                                                                            TIM00730
VAR (42): CALL READ_IT;                                                     SIM03260
          A = BIT (BINARY (A,15,0) -                                        SIM03270
                   BINARY (BOOL (Y_BAR,Q,'0001'),15,0),30);                 SIM03280
          CALL J_CHANGE;                                                    SIM03290
          GO TO ENDING;                                                     SIM03300
                                                                            SIM03310
   /*     THIS INSTRUCTION SUBTRACTS FROM THE CONTENTS OF THE               TIM00800
          A REGISTER THE BIT BY BIT PRODUCT OF THE OPERAND                  TIM00810
          Y_BAR AND THE CONTENTS OF THE Q REGISTER. THE ORIGINAL            TIM00820
          CONTENTS OF THE A & Q REGISTERS ARE SAVED OFF IN THE              TIM00830
          VARIABLES A1 AND Q1. THE CONTENTS OF THE ORIGINAL                 TIM00840
          REGISTERS ARE NOT DISTURBED BY THIS INSTRUCTION.                  TIM00850
          NOTE THE USE OF THE PL/1 FUNCTIONS (BIT,BOOL,BINARY).     */      TIM00860
                                                                            TIM00870
VAR (43): CALL READ_IT;                                                     SIM03320
          A1 = A;                                                           TIM00750
          Q1 = Q;                                                           TIM00760
          A = BIT (BINARY (A,15,0) -                                        SIM03330
                   BINARY (BOOL (Y_BAR,Q,'0001'),15,0),30);                 SIM03340
          CALL J_CHANGE;                                                    SIM03350
          A = A1;                                                           TIM00770
          Q = Q1;                                                           TIM00780
          GO TO ENDING;                                                     SIM03360
                                                                            SIM03370
   /*     THIS INSTRUCTION ENTERS INTO THE A REGISTER THE BIT               TIM00900
          BIT PRODUCT OF THE OPERAND Y AND THE CONTENTS OF THE              TIM00910
          Q REGISTER. THE PARITY OF THE A REGISTER IS THEN CHECKED          TIM00920
          BY THE CALL TO THE PROCEDURE (PAR_CHECK). IF THE PARITY           TIM00930
          EVEN AND THE VALUE OF THE J DESIGNATOR IS 2 OR THE                TIM00940
          PARITY IS ODD AND THE VALUE OF THE J DESIGNATOR IS 3              TIM00950
          THEN SKIP TO THE NEXT INSTRUCTION. OTHERWISE STORE THE            TIM00960
          CONTENTS OF THE A REGISTER AT STORAGE ADDRESS Y AS                TIM00970
          MODIFIED BY THE K DESIGNATOR. THE STORING IS ACCOMPLISHED         TIM00980
```

```
          BY THE CALL TO THE PROCEDURE (REP_STO).                    */  TIM00990
                                                                         TIM01000
VAR (44): A = BOOL (MEMORY (VAL_15 (Y)),Q,'0001');                       SIM03380
          PARITY = PAR_CHECK (A);                                        SIM03390
          J_DEC = OCTAL (J);                                             TIM01010
          IF J_DEC = 2 & PARITY = 'EVEN' THEN GO TO SKIP;                TIM01020
          IF J_DEC = 3 & PARITY = 'ODD ' THEN GO TO SKIP;                TIM01030
         R_DEC = OCTAL (R);
         IF R_DEC > 4 & FLAG = '1'B THEN
          Y = BIT (BINARY (Y,15,0) + BINARY (B_REGIS (6),15,0),15);
          CALL REP_STO (A,Y);                                            SIM03420
          GO TO ENDING;                                                  SIM03430
                                                                         SIM03440
    /*    THIS INSTRUCTION ADDS TO THE CONTENTS OF THE A                 TIM01070
          REGISTER THE BIT BY BIT PRODUCT OF THE OPERAND                 TIM01080
          Y AND THE CONTENTS OF THE Q REGISTER. THE CONTENTS             TIM01090
          OF THE A REGISTER ARE THEN STORED AT STORAGE                   TIM01100
          ADDRESS Y AS MODIFIED BY THE K DESIGNATOR IN                   TIM01110
          THE CALL TO THE PROCEDURE (REP_STO).                     */    TIM01120
                                                                         TIM01130
VAR (45): A = BIT (BINARY (A,15,0) +                                     SIM03450
                   BINARY (BOOL (Y,Q,'0001'),15,0),30);                  SIM03460
         R_DEC = OCTAL (R);
         IF R_DEC > 4 & FLAG = '1'B THEN
          Y = BIT (BINARY (Y,15,0) + BINARY (B_REGIS (6),15,0),15);
          CALL REP_STO (A,Y);                                            SIM03470
          CALL J_CHANGE;                                                 SIM03480
          GO TO ENDING;                                                  SIM03490
                                                                         SIM03500
    /*    THIS INSTRUCTION SUBTRACTS FROM THE CONTENTS OF                TIM01160
          THE A REGISTER THE BIT BY BIT PRODUCT OF THE                   TIM01170
          OPERAND Y AND THE CONTENTS OF THE Q REGISTER. THE              TIM01180
          RESULT IS THEN STORED AT STORAGE ADDRESS Y AS                  TIM01190
          MODIFIED BY THE K DESIGNATOR IN THE CALL TO                    TIM01200
          THE PROCEDURE (REP_STO).                                 */    TIM01210
                                                                         TIM01220
VAR (46): A = BIT ( BINARY (A,15,0) -                                    SIM03510
                    BINARY (BOOL (MEMORY (VAL_15 (Y)),Q,'0001'),15,0),30); SIM03520
         R_DEC = OCTAL (R);
         IF R_DEC > 4 & FLAG = '1'B THEN
          Y = BIT (BINARY (Y,15,0) + BINARY (B_REGIS (6),15,0),15);
          CALL REP_STO (A,Y);                                            SIM03530
          CALL J_CHANGE;                                                 SIM03540
          GO TO ENDING;                                                  SIM03550
                                                                         SIM03560
    /*    THIS INSTRUCTION STORES AT ADDRESS Y AS MODIFIED               TIM01260
          BY THE K DESIGNATOR IN THE CALL TO PROCEDURE                   TIM01270
          (REP_STO) THE BIT BY BIT PRODUCT OF THE                        TIM01280
```

```
          CONTENTS OF THE A AND Q REGISTERS.                          */

VAR (47): CALL STORE (BOOL (A,Q,'0001'),Y);
          CALL J_CHANGE;   GO TO ENDING;

   /*   THIS INSTRUCTION SETS THE BITS OF A TO 1 BITS OF
        OPERAND Y VIA PL/1 FUNCTION BOOL                              */

VAR (50): CALL READ_IT;      A = BOOL (A,Y_BAR,'0111');
          CALL J_CHANGE;     GO TO ENDING;

  /*  THIS INSTRUCTION COMPLIMENTS THE BITS OF A TO 1 BITS
      CORRESPONDING TO 1 BITS IN Y  VIA PL/1 FUNCTION BOOL            */

VAR (51): CALL READ_IT;    A = BOOL (A,Y_BAR,'0100');
          CALL J_CHANGE;     GO TO ENDING;

  /*  THIS INSTRUCTION SETS THE BITS OF A TO 0 CORRESPONDING
      TO THE 1 BITS IN Y VIA THE PL/1 FUNCTION BOOL                   */

VAR (52): CALL READ_IT;     A = BOOL (A,Y_BAR,'0010');
          CALL J_CHANGE;    GO TO ENDING;

  /*  THIS INSTRUCTION SETS THE BITS OF A TO BITS OF Y
      CORRESPONDING TO THE 1 BITS OF Q VIA THE PL/1 BOOL              */

VAR (53): CALL READ_IT;
          A = BOOL (A,BOOL (Y_BAR,Q,'0001'),'0111');
          CALL J_CHANGE;    GO TO ENDING;

  /*      THIS INSTRUCTION SETS THE INDIVIDUAL BITS OF A TO 1
          CORRESPONDING TO THE U  1 BITS OF THE OPERAND Y.
           IT THEN STORES THE CONTENTS OF REGISTER A INTO
          LOCATION Y. THE CALL TO REP STO ACCOMLIISHES THE STORING   */

VAR (54): A = BOOL (A,MEMORY (VAL_15 (Y)),'0111');
```

```
          CALL J_CHANGE;                                                    SIM03820
         R_DEC = OCTAL (R);
         IF R_DEC > 4 & FLAG = '1'B THEN
          Y = BIT (BINARY (Y,15,0) + BINARY (B_REGIS (6),15,0),15);
          CALL REP_STO (A,Y);                                               SIM03830
          GO TO ENDING;                                                     SIM03840
                                                                            SIM03850
   /*     THIS INSTRUCTION COMPLIMENTS THE BITS OF A CORRESPONDING
          TO THE 1 BITS OF THE OPERAND Y AND THEN STORES THE
          CONTENTS OF A AT LOCATION Y VIA THE CALL TO REPSTO.       */

VAR (55): A = BOOL (A,MEMORY (VAL_15 (Y)),'0100');                          SIM03860
          CALL J_CHANGE;                                                    SIM03870
         R_DEC = OCTAL (R);
         IF R_DEC > 4 & FLAG = '1'B THEN
          Y = BIT (BINARY (Y,15,0) + BINARY (B_REGIS (6),15,0),15);
          CALL REP_STO (A,Y);                                               SIM03880
          GO TO ENDING;                                                     SIM03890
                                                                            SIM03900
   /*     THIS INSTRUCTION SETS THE BITS OF A TO 0 CORRESPONDING
          TO THE 1 BITS OF THE OPERAND Y AND THEN STORES THE
          CONTENTS OF A AT THE LOCATION Y VIS THE CALL TO REP STO.  */
VAR (56): A = BOOL (A,MEMORY (VAL_15 (Y)),'0010');                          SIM03910
          CALL J_CHANGE;                                                    SIM03920
         R_DEC = OCTAL (R);
         IF R_DEC > 4 & FLAG = '1'B THEN
          Y = BIT (BINARY (Y,15,0) + BINARY (B_REGIS (6),15,0),15);
          CALL REP_STO (A,Y);                                               SIM03930
          GO TO ENDING;                                                     SIM03940
                                                                            SIM03950
    /*    THIS INSTRUCTION SETS THE BITS OF A TO 0 CRRRESPONDING
          TO THE 1 BITS OF Q.  IT THEN FORMS THE BIT BY BIT
         PRODUCT OF Q AND THE OPERAND Y AND SETS THE BITS OF A
          TO 1 CORRESPONDING TO THE 1 BUTS OF THE PRODUCT. IT
          THEN STORES THE CONTENTS OF A AT LOCATION Y,               */

VAR (57): A = BOOL (A,Q,'0010');                                            SIM03960
          A = BOOL (A,BOOL (MEMORY (VAL_15 (Y)),Q,'0001'),'0111');          SIM03970
          CALL J_CHANGE;                                                    SIM03980
         R_DEC = OCTAL (R);
         IF R_DEC > 4 & FLAG = '1'B THEN
          Y = BIT (BINARY (Y,15,0) + BINARY (B_REGIS (6),15,0),15);
          CALL REP_STO (A,Y);                                               SIM03990
          GO TO ENDING;                                                     SIM04000
                                                                            SIM04010
   /*     THIS INSTRUCTION IS AN ARITHMETIC JUMP TO THE
          K MODIFIED Y OPERAND ADDRESS AS DETERMINED BU THE J
          DESIGNATOR FOR VARIOUS ARITHMETIC COMPARISONS OF THE
```

```
          A AND Q REGISTERS.                                          */
VAR (60): CALL READ_IT;                                               SIM04020
          J_DEC = OCTAL (J);                                          LES00040
          IF J_DEC = 0 THEN GO TO ENDING;                             LES00050
          IF J_DEC = 1    |                                           LES00060
             J_DEC = 2    &  Q > 0   |                                LES00070
             J_DEC = 3    & Q < 0    |                                LES00080
             J_DEC = 4    &  A = 0   |                                LES00090
             J_DEC = 5    &  A ¬= 0  |                                LES00100
             J_DEC = 6    &  A > 0   |                                LES00110
             J_DEC = 7    &  A < 0                                    LES00120
                THEN DO;                                              LES00130
                P = SUBSTR (Y_BAR,16);                                LES00140
                P_OCT = VAL_15 (P);                                   LES00150
                P_DEC = P;                                            LES00160
                GO TO START2;                                         LES00170
                END;                                                  LES00180
          GO TO ENDING;                                               SIM04030
                                                                      SIM04040
   /*     THIS INSTRUCTION JUMPS TO THE K MODIFIED Y OPERAND
          ADDRESS FOR THE SPECIAL CONDITIONS OF THE J DESIGNATOR
          AND THE JUMP AND STOP MACHINE CONSOLE SWITCHES. THESE
          SWITCHES ARE SIMULATED AND ARE SET TO 0 INITIALLY.          */

VAR (61): CALL READ_IT;                                               SIM04050
          J_DEC = OCTAL (J);                                          LES00220
          IF J_DEC = 0    |                                           LES00230
             J_DEC = 1 & JUMP1  |                                     LES00240
             J_DEC = 2 & JUMP2  |                                     LES00250
             J_DEC = 3 & JUMP3  |                                     LES00260
             J_DEC > 3                                                LES00270
                  THEN DO;                                            LES00280
                  P = SUBSTR (Y_BAR,16);                              LES00290
                  P_OCT = VAL_15 (P);                                 LES00300
                  P_DEC = P;                                          LES00310
                  END;                                                LES00320
          IF J_DEC = 4       |                                        LES00330
             J_DEC = 5 & STOP5  |                                     LES00340
             J_DEC = 6 & STOP6  |                                     LES00350
             J_DEC = 7 & STOP7                                        LES00360
                THEN GO TO FINISH;                                    LES00370
          GO TO ENDING;                                               SIM04060
                                                                      SIM04070
   /*     THIS INSTRUCTION IS DESIGNED TO JUMP TO A NEW PROGRAM
          ADDRESS AS DETERMINED BY THE J HAT INPUT CHANNEL AND
          THE INPUT BUFFER. IT IS TOTALLY IGNORED IN THIS
          TRANSLATION PACKAGE.                                        */
```

```
VAR (62): CALL READ_IT;                                                 SIM04080
          GO TO ENDING;                                                 SIM04090
                                                                        SIM04100
   /*     THIS INSTRUCTION IS DESIGNED TO JUMP TO A NEW PROGRAM
         ADDRESS AS DETERMINED BY THE ACTIVITY OF THE OUTPUT
          CHANNEL DESIGNATED BY J HAT. THIS INSTRUCTION
          IS TOTALLY IGNORED IN THIS TRANSLATION.                  */

VAR (63): CALL READ_IT;                                                 SIM04110
          GO TO ENDING;                                                 SIM04120
                                                                        SIM04130
  /*      THIS INSTRUCTION EXECUTES A JUMP TO THE OPERAND Y + 1
          FOR J DESIGNATED COMPARISONS WITH THE A AND Q REGISTERS.
          THE PRESENT INSTRUCTION ADDRESS IS SAVED IN ADDRESS
          OF THE K MODIFIED OPERAND Y;.                           */

VAR (64): CALL READ_IT;                                                 SIM04140
     J_DEC = OCTAL (J);
    IF J_DEC = 0 THEN GO TO ENDING;
    IF J_DEC = 1     |
      J_DEC = 2  & Q > 0   |
      J_DEC = 3 & Q < 0    |
      J_DEC = 4   & A = 0   |
       J_DEC = 5 & A ¬= 0   |
      J_DEC = 6 & A > 0    |
      J_DEC = 7 & A < 0    THEN DO;
    HOLD4 = VAL_THIRTY (Y_BAR);
    IF HOLD4 > 32767 THEN
         SUBSTR ( MEM2 (HOLD4 - 32767),16,15) = P;
      ELSE SUBSTR (MEM1 (HOLD4),16,15) = P;
         P = BIT (HOLD4,15);
      END;
          GO TO ENDING;                                                 SIM04150
                                                                        SIM04160
  /*      THIS INSTRUCTION IS DESIGNED TO JUMP TO A NEW PROFRAM
          ADDRESS BASED UPON THE J DESIGNATOR AND THE CONSOLE
          SWITCH SETTINGS. THIS INSTRUCTION IS IGNORED.           */

VAR (65): CALL READ_IT;                                                 SIM04170
          GO TO ENDING;                                                 SIM04180
                                                                        SIM04190
  /*    THIS INSTRUCTION IS DESIGNED TO MANIPULATE INPUT BUFFERS,
          CHANNELS, ETC. ON THE CP 901 AND IS NOT APPLICABLE
          FOR THIS TRANSLATION.                                   */

VAR (66):PUT FILE (SYSPRINT) EDIT ('INTERRUPT BUFFER') (SKIP,A);
          GO TO ENDING;                                                 SIM04210
```

```
                                                                      SIM04220
   /*      THIS INSTRUCTION IS DESIGNED TO MANIPULATE OUTPUT
           BUFFERS AND CHANNELS ON THE CP 901 AND IS NOT
           APPLICABLE FOR THIS TRANSLATION.                          */

 VAR (67):PUT FILE (SYSPRINT) EDIT ('VAR LABEL 67') (SKIP,A);
           GO TO ENDING;                                              SIM04240
                                                                      SIM04250
    /*     THIS INSTRUCTION SETS B REGISTER # 7 TO THE
           LOWER FIFTEEN BITS OF THE K MODIFIED OPERAND Y.
           IF THE LOWER 15 BITS OF Y ARE NON ZERO THEN REPEAT THE
           NEXT INSTRUCTIONFOR THE COUNTER IN THE LOWER 15
           BITS OF Y. THE J DESIGNATOR IS NOW THE R DESIGNATOR.
           IT HAS THE FOLLOWING SPECIAL MODS:
                R = 0 OPERAND IS CONSTANT
                R = 1 OPERAND INCREMENTED BY + 1 EACH EXECUTION
                R = 2 OPERAND DECREASED BY 1 EACH EXECUTIOIN;
                R = 3  REPEAT INITIAL B REGISTER MODIFICATION
    N                FOR EACH EXECUTION
                R = 4   MODIFI ONLY REPLACE INSTRUCTION BY    B
                   REGISTER (6) FOR STORING
                R = 5   ADD 1 TO OPERAND ADDRESS MODIFY REPLACE
                INSTRUCTION BY B REGISTER (6) FOR STORING
                R = 6   SUBTRACT 1 FROM OPERAND ADDRESS  MODIFY
                REPLACE INSTRUCTIONS BY B REGISTER (6) FOR STROING
                  R = 7  REPEAT ORIGINAL B REGISTER MODIFICATION
                    AND MODIFY REPLACE INSTRUCTIONS BY B REGISTER
                   (6) FOR STORING;
              REPLACE INSTRUCTIONS ARE : 24,25,34-37,44-46,54-57.    */

 VAR (70): CALL READ_IT;                                              SIM04260
           IF Y_BAR = 0 THEN DO;
             P_DEC = P_DEC + 1;
             P = BIT (P_DEC, 15);
           END;
           ELSE DO;
           B_REGIS (7) = SUBSTR (Y_BAR,16,15);
           R = J;
           R_DEC = OCTAL (R);
           HOLD5 = VAL_15 (B_REGIS (7));
           HOLD2 = P_DEC + 1;
           HOLD1 = 0;
 REPEAT:   P_DEC = HOLD2;
           HOLD1 = HOLD1 + 1;
DO L = HOLD1 TO HOLD5;
           FLAG = '1'B;
           P = BIT (P_DEC,15);
           GO TO START2;
```

```
          END;
          END;
           GO TO ENDING;                                                  SIM04280
                                                                          SIM04290
   /*     THIS INSTRUCTION SKIPS THE NEXT INSTRUCTION IF THE
          B REGISTER DESIGNATED BY J IS EQUAL TO THE LOWER 15
          BITS OF THE K MODIFIED Y OPERAND. IN THIS CASE THAT
          B REGISTER IS ZEROED. ELSE EXECUTE THE NEXT INSTRUCTION
          AND INCREMENT THE DESIGNATED B REGISTER BY 1.              */

VAR (71): CALL READ_IT;                                                   SIM04300
          J_DEC = OCTAL (J);
          IF B_REGIS (J_DEC) = SUBSTR (Y_BAR,16,15) THEN DO;
            B_REGIS (J_DEC) = 0;
            P_DEC = P_DEC + 1;
            P = BIT (P_DEC, 15);
          END;
          ELSE B_REGIS (J_DEC) = BIT (BINARY (B_REGIS (J_DEC),15,0) +
                                      1,15);
           GO TO ENDING;                                                  SIM04310
                                                                          SIM04320
   /*     THIS INSTRUCTION EXECUTES A JUMP TO THE K MODIFIED
          OPERAND Y ADDRESS IF THE J DESIGNATED B REGISTER IS
          NOT ZERO. (REDUCE IT BY 1 IN THIS CASE).                   */

VAR (72): CALL READ_IT;                                                   SIM04340
           IF BINARY (B_REGIS (J),15,0) ¬= 0 THEN DO;                     SIM04350
                B_REGIS (J) = BIT (BINARY (B_REGIS (J),15,0) -            SIM04360
                      1,15);
                P = VAL_15 (SUBSTR (Y_BAR,16,15));                        SIM04380
                GO TO START2;                                             SIM04390
                END;                                                      SIM04400
           GO TO ENDING;                                                  SIM04410
                                                                          SIM04420
    /*    THIS INSTRUCTION ESTABLISHES AN INPUT BUFFER AS
          DETERMINED BY THE K HAT DESIGNATOR MODIFYING THE
          Y OPERAND. ADDRESS 100 PLUS J HAT (HOLD3) CONTAINS
          THE SCOPE OF THE AREA FOR THE READING. READING WILL
          CONTINUE UNTIL THE UPPER HALF OF THE LOCATION (HOLD3)
          AND THE LOWER HALF ARE EQUAL. IF K HAT = 0 Y IS
          STORED IN THE LOWER 15 BITS OF THE LOAATION (HOLD3).
          IF KA HAT = 1 THE LOWER 15 BITS OF THE LOCATIOM OF (
          Y AEE STORED INTO THE LOWER 15 BITS OF (HOLD3).
          K HAT = 2 IS NOT ALLOWED,  IF K HAT = 3 LOCATION
          (HOLD3) IS SET TO THE LOCATION OF THEOPERAND Y.             */

VAR (73):PUT FILE (SYSPRINT) EDIT ('VAR LABEL 73') (SKIP,A);
          K_DEC = OCTAL ('0'B || K_HAT);
```

```
            ASTR = MEMORY (VAL_15 (Y));
            J_DEC = OCTAL ('00'B || SUBSTR (J_HAT,1,1)) * 10 +
                 OCTAL (SUBSTR (J_HAT,2,3));
            HOLD3 = 100 + J_DEC;
            SUBSTR (MEM1 (HOLD3),16,15) = Y;
            IF K_DEC = 1 THEN
            SUBSTR (MEM1 (HOLD3),16,15) = SUBSTR (ASTR,16,15);
            IF K_DEC = 3 THEN
            MEM1 (HOLD3) = ASTR;
            DO WHILE (SUBSTR (MEM1 (HOLD3),1,15) ¬=
            SUBSTR (MEM1 (HOLD3),16,15));
            ON ENDFILE (SYSIN) BEGIN;
            PUT FILE (SYSPRINT) EDIT ('EXPECTED MORE INPUT ') (SKIP,A);
            GO TO ENDING;
            END;
            GET FILE (SYSIN) EDIT (INPUT1) (A(30));
            MEM1 (HOLD3) = BIT (BINARY (MEM1 (HOLD3)) + 1);
            END;
             GO TO ENDING;                                                  SIM04440
                                                                            SIM04450
   /*       THIS INSTRUCTION ESTABLISHES AN OUTPUT BUFFER USING
            AN AREA DEFINED BY THE K HAT MODIFIED Y OPERAND.
            IF K HAT = 2 THIS IS AN EXTERNAL FUNCTION BUFFER AND
            IS NOT IMPLEMENTED IN THIS TRANSLATOR.  J HAT
            DESIGNATES THE DATA CHANNEL AND IS ALSO NOT IMPLEMENTED.
            ADDRESS 120 PLUS J HAT IS THE CONTROL ADDRESS (HOLD3).
            IF K HAT = 0 THEN THE LOWER 15 BITS OF LOCATION
            (HOLD3) = Y;
            IF K HAT = 1 THEN THE LOWER 15 BITS OF LOCATION
            (HOLD3) = LOWER 15 BITS OF LOCATION (Y). IF K HAT
            = 3 THEN THE LOCATION (HOLD3) = LOCATION IN (Y).
            OUTPUT WILL CONTINUE STARTING AT THE INITIAL ADDRESS
            OF THE LOWER 15 BITS OF LOCATION (HOLD3) UNTIL THE
            UPPER 15 BITS = LOWER 15 BITS.                                  */

VAR (74):PUT FILE (SYSPRINT) EDIT ('VAR LABEL 74') (SKIP,A);
            K_DEC = OCTAL ('0'B || K_HAT);
            J_DEC = OCTAL ('00'B || SUBSTR (J_HAT,1,1)) * 10 +
            OCTAL (SUBSTR (J_HAT,2,3));
            HOLD3 = 120 + J_DEC;
            ASTR = MEMORY (VAL_15 (Y));
            IF K_DEC = 0 THEN
                 SUBSTR (MEM1 (HOLD3),16,15) = Y;
            IF K_DEC = 1 THEN
            SUBSTR (MEM1 (HOLD3),16,15) = SUBSTR (ASTR,16,15);
            IF K_DEC = 3 THEN
            MEM1 (HOLD3) = ASTR;
          DO WHILE ( SUBSTR (MEM1 (HOLD3),1,15) ¬= SUBSTR (MEM1 (HOLD3),
```

```
                    16,15));
          HOLD5 = VAL_15 (SUBSTR (MEM1 (HOLD3),16,15));
          PUT FILE (SYSPRINT) EDIT ('MEMORY  ', HOLD5, ' IS ',
                    MEMORY (HOLD5)) (SKIP,A,F(8),A,B);
  SUBSTR (MEM1 (HOLD3),16,15) = BIT (BINARY (SUBSTR (MEM1 (HOLD3),
          16,15),15,0) + 1,15);
          END;
           GO TO ENDING;                                                SIM04470
                                                                       SIM04480
    /*    THIS INSTRUCTION ESTABLISHES AN INPUT BUFFER WITH
          MONITOR AND IS THEREFORE NOT IMPLEMENTED.              */

VAR (75):PUT FILE (SYSPRINT) EDIT ('VAR LABEL 75') (SKIP,A);
           GO TO ENDING;                                               SIM04500
                                                                       SIM04510
     /*   THIS INSTRUCTION ESTABLISHES AN OUTPUT BUFFER WITH
          MONITOR AND IS THEREFORE NOT IMPLEMENTED.              */

VAR (76):PUT FILE (SYSPRINT) EDIT ('VAR LABEL 76') (SKIP,A);
           GO TO ENDING;                                               SIM04530
                                                                       SIM04540
    /*    THIS INSTRUCTION IS THE SERIES OF THE DIRECT
          ADDRESSING MODE FUNCTOON. VARIABLE FS NOW HAS THE
          OPCODE FOR THE FOLLOWING FUNCTIONS.                    */

VAR (77):PUT FILE (SYSPRINT) EDIT ('VAR LABEL 77') (SKIP,A);
          F_DEC = OCTAL (SUBSTR (FS,1,3)) * 10 +
                  OCTAL (SUBSTR (FS,4,3));
          IF F_DEC = 11 THEN GO TO VAR_11;
          IF F_DEC = 15 THEN GO TO VAR_15;
          IF F_DEC = 44 THEN GO TO VAR_44;
          IF F_DEC = 50 THEN GO TO VAR_50;
          IF F_DEC = 54 THEN GO TO VAR_54;
          IF F_DEC = 63 THEN GO TO VAR_63;
          IF F_DEC = 64 THEN GO TO VAR_64;
          IF F_DEC = 65 THEN GO TO VAR_65;
           GO TO ENDING;                                               SIM04560
                                                                       SIM04570
    /*    THIS INSTRUCTION LOADS THE A REGISTER WITH THE MEMORY
          ADDRESS OF THE LEAST SIGNIFICANT 17 BITS OF THE U (Y2)   */

 VAR_11:   HOLD4 = VAL_15 (SUBSTR (Y_2,3));
          HOLD4 = HOLD4 + (1000000 * OCTAL ('0'B || SUBSTR (Y_2,1,2)));
          A = MEMORY (HOLD4);
          GO TO ENDING;

    /*    THIS INSTRUCTION IS A DIRECT ADDRESSING INSTRUCTION
          AND STORES THE CONTENTS OF THE A REGISTER IB LOCATION
```

```
          DETERMINED BY THE LEAST SIGNIFICANT 17 BITS OF THE
          U REGISTER (Y_2).                                          */

VAR_15:   HOLD4 = VAL_15 (SUBSTR (Y_2,3));
          HOLD4 = HOLD4 + (1000000 * OCTAL ('0'B || SUBSTR (Y_2,1,2)));
          IF HOLD4 > 32767 THEN
          MEM2 ( HOLD4 - 32767) = A;
          ELSE MEM1 (HOLD4) = A;
          GO TO ENDING;

 /*       THIS INSTRUCTION TESTS THE MODIFIED Y OPERAND FOR 0
          IF IT IS 0 THEN NEXT INSTRUCTION IS SKIPPED.               */

VAR_44:   IF Y = 0 THEN DO;
          P_DEC = P_DEC + 1;
          P = BIT (P_DEC,15);
          END;
          GO TO ENDING;

 /*       THIS INSTRUCTION IS DIRECT ADDRESSING AND STORES THE
          SEVEN LEAST SIGNIFICANT BITS OF K HAT MODIFIED Y OPERAND
          INTO ABSOLUTE PAGE REGISTER DESIGNATED BY J HAT AND IS
          THUS NOT IMPLIMENTED.                                      */

VAR_50:   GO TO ENDING;
 /*       THIS INSTRUCTION IN DIRECT ADDRESSING MODE STORES
          BITS OF ABSOLUTE PAGE REGISTER DESIGNATED BY J HAT
          INTO THE K HAT MODIFIED Y OPERAND. IT IS NOT IMPLEMENTED.  */

VAR_54:   GO TO ENDING;

 /*       THIS DIRECT ADDRESSING INSTRUCTION LOADS THE VALUE OF THE
          P REGISTER INTO THE B 7 REGISTER AND JUMPS TO LOWER 15
          BITS OF THE Y MODIFIED OPERAND TO SAVE PROGRMAS IN SHARED
          MEMORY. (IT IS NOT IMPLEMENTED).                           */

VAR_63:  GO TO ENDING;

 /*       SAME AS 77 63 BUT DIRECT MODE SET                          */

VAR_64:   GO TO ENDING;

 /*    SAME AS 77 63 BUT WITH PAGING MODE SET    */

VAR_65:   GO TO ENDING;

ENDING:  IF FLAG = '1'B THEN DO;
          FLAG = '0'B;
```

```
          GO TO REPEAT;                                                    SIM04590
          END;                                                             SIM04600
          P_DEC = P_DEC + 1;                                               SIM04610
         P = BIT (P_DEC,15);                                               SIM04620
         GO TO START2;                                                     SIM04630
                                                                           SIM04640
OCTAL: PROCEDURE (INP) FIXED BINARY;                                       SIM04650
DCL INP BIT (3);                                                           SIM04660
IF INP = '000' THEN RETURN (0);                                            SIM04670
IF INP = '001' THEN RETURN (1);                                            SIM04680
IF INP = '010' THEN RETURN (2);                                            SIM04690
IF INP = '011' THEN RETURN (3);                                            SIM04700
IF INP = '100' THEN RETURN (4);                                            SIM04710
IF INP = '101' THEN RETURN (5);                                            SIM04720
IF INP = '110' THEN RETURN (6);                                            SIM04730
RETURN (7);                                                                SIM04740
END OCTAL;                                                                 SIM04750
                                                                           SIM04760
REP_STR: PROCEDURE (ABIT,ANUM);                                            SIM04770
PUT FILE(SYSPRINT) EDIT ('IN PROC REP_STR  ') (SKIP,A);                    SIM04780
          DCL ABIT BIT (1);                                                SIM04790
          DCL ANUM FIXED BINARY (15);                                      SIM04800
          ASTR = '';                                                       SIM04810
OVER:  IF ANUM = 0 THEN GO TO STOPIT;                                      SIM04820
       ELSE DO ;                                                           SIM04830
       ASTR = ASTR || ABIT;                                                SIM04840
       ANUM = ANUM - 1;                                                    SIM04850
       GO TO OVER;                                                         SIM04860
       END;                                                                SIM04870
STOPIT: END REP_STR;                                                       SIM04880
                                                                           SIM04890
VAL_THIRTY: PROCEDURE (REGIS) FIXED BINARY (15);                           SIM04900
PUT FILE(SYSPRINT) EDIT ('IN PROC VAL THIRTY')(SKIP,A);                    SIM04910
             DCL REGIS BIT (30);                                           SIM04920
             RETURN (OCTAL (SUBSTR (REGIS,1,3))  * (10 ** 9) +             SIM04930
                     OCTAL (SUBSTR (REGIS,4,3))  * (10 ** 8) +             SIM04940
                     OCTAL (SUBSTR (REGIS,7,3))  * (10 ** 7) +             SIM04950
                     OCTAL (SUBSTR (REGIS,10,3)) *  1000000  +             SIM04960
                     OCTAL (SUBSTR (REGIS,13,3)) *   100000  +             SIM04970
                     OCTAL (SUBSTR (REGIS,16,3)) *    10000  +             SIM04980
                     OCTAL (SUBSTR (REGIS,19,3)) *     1000  +             SIM04990
                     OCTAL (SUBSTR (REGIS,22,3)) *      100  +             SIM05000
                     OCTAL (SUBSTR (REGIS,25,3)) *       10  +             SIM05010
                     OCTAL (SUBSTR (REGIS,28,3))    );
END VAL_THIRTY;

VAL_15: PROCEDURE (REGIS) FIXED BINARY (15);
PUT FILE(SYSPRINT) EDIT ('IN PROC VAL 15    ')(SKIP,A);
```

```
        DCL REGIS BIT (15);                                                    SIM05020
        RETURN (OCTAL (SUBSTR (REGIS,1,3)) * 10000 +                           SIM05030
                OCTAL (SUBSTR (REGIS,4,3)) *  1000 +                           SIM05040
                OCTAL (SUBSTR (REGIS,7,3)) *   100 +                           SIM05050
                OCTAL (SUBSTR (REGIS,10,3)) *   10 +                           SIM05060
                OCTAL (SUBSTR (REGIS,13,3))    );                              SIM05070
END VAL_15;                                                                    SIM05080
                                                                               SIM05090
CIR_SFT:   PROCEDURE (STRNG,VALUE) BIT (30);                                   SIM05100
PUT FILE(SYSPRINT) EDIT ('IN PROC  CIR SFT  ')(SKIP,A);
           DCL STRNG BIT (30);                                                 SIM05120
           DCL VALUE FIXED BIN (15);                                           SIM05130
       CS: IF VALUE = 0 THEN RETURN (STRNG);                                   SIM05140
           ELSE DO;                                                            SIM05150
                STRNG = SUBSTR (STRNG,2,29) || SUBSTR (STRNG,1,1);             SIM05160
                VALUE = VALUE - 1;                                             SIM05170
                GO TO CS;                                                      SIM05180
           END;                                                                SIM05190
END CIR_SFT;                                                                   SIM05200
                                                                               SIM05210
STORE: PROCEDURE (BITS,BATS);                                                  SIM05220
PUT FILE(SYSPRINT) EDIT ('IN PROC STORE     ')(SKIP,A);
       DCL BITS BIT (30);                                                      SIM05230
       DCL BATS BIT (15);                                                      SIM05240
       HOLD4 = VAL_15 (BATS);                                                  SIM05250
        IF K = 0 THEN Q = BITS;                                                SIM05260
        IF K = 1 THEN IF HOLD4 > 32767 THEN MEM2 (HOLD4 - 13516) =             SIM05270
                                  SUBSTR (MEM2 (HOLD4 - 13516),1,15) ||        SIM05280
                                  SUBSTR (BITS,16,15);                         SIM05290
                      ELSE MEM1 (HOLD4) = SUBSTR (MEM1 (HOLD4),1,15) ||        SIM05300
                                          SUBSTR (BITS,16,15);                 SIM05310
        IF K = 2 THEN IF HOLD4 > 32767 THEN MEM2 (HOLD4 - 13516) =             SIM05320
                           SUBSTR (BITS,16,15) ||                              SIM05330
                            SUBSTR (MEM2 (HOLD4 - 13516),16,15);               SIM05340
                         ELSE MEM1 (HOLD4) = SUBSTR (BITS,16,15) ||            SIM05350
                                             SUBSTR (MEM1 (HOLD4),16,15);      SIM05360
        IF K = 3 THEN IF HOLD4> 32767 THEN MEM2 (HOLD4 - 13516) = BITS;        SIM05370
                      ELSE MEM1 (HOLD4) = BITS;                                SIM05380
        IF K = 4 THEN A = BITS;                                                SIM05390
         IF K = 5 THEN IF HOLD4 > 32767 THEN
         SUBSTR (MEM2 (HOLD4 - 32767),16,15) = ¬ (BITS);
         ELSE SUBSTR (MEM1 (HOLD4),16,15) = ¬ (BITS);
         IF K = 6 THEN IF HOLD4 > 32767 THEN
         SUBSTR (MEM2 (HOLD4 - 32767),1,15) = ¬ (BITS);
         ELSE SUBSTR (MEM1 (HOLD4),1,15) = ¬ (BITS);
         IF K = 7 THEN IF HOLD4 > 32767 THEN
         MEM2 (HOLD4 - 32767) = ¬ (BITS);
         ELSE MEM1 (HOLD4) = ¬ (BITS);
```

```
        END STORE;                                                          SIM05400
                                                                            SIM05410
READ_IT: PROCEDURE;                                                         SIM05420
PUT FILE(SYSPRINT) EDIT ('IN PROC S READ IT ')(SKIP,A);
          DCL VAL_K FIXED BINARY (15);                                      SIM05430
          VAL_K = OCTAL (K);                                                SIM05450
          CALL REP_STR ('0',15);                                            SIM05470
          IF VAL_K = (0) THEN Y_BAR = ASTR || Y;                            SIM05480
          IF VAL_K = (1) THEN Y_BAR = ASTR ||                               SIM05490
                                      SUBSTR (MEMORY (VAL_15 (Y)),16,15);   SIM05500
          IF VAL_K = 2 THEN Y_BAR = ASTR ||                                 SIM05510
                                      SUBSTR (MEMORY (VAL_15 (Y)),1,15);    SIM05520
          IF VAL_K = 3 THEN Y_BAR = MEMORY (VAL_15 (Y));                    SIM05530
          CALL REP_STR (SUBSTR (Y_BAR,16,1),15);                            SIM05540
          IF VAL_K = 4 THEN Y_BAR = ASTR || Y;                              SIM05550
          IF VAL_K = 5 THEN Y_BAR = ASTR ||                                 SIM05560
                                    SUBSTR (MEMORY (VAL_15 (Y)),16,15);     SIM05570
          IF VAL_K = 6 THEN DO;                                             SIM05580
                CALL REP_STR (SUBSTR (Y_BAR,1,1),15);                       SIM05590
                Y_BAR = ASTR || SUBSTR (MEMORY (VAL_15 (Y)),1,15);          SIM05600
          END;                                                              SIM05610
          IF F_DEC = 22 | F_DEC = 52 | F_DEC = 53 THEN GO TO OMT;           SIM05620
          IF VAL_K = 7 THEN Y_BAR = MEMORY (VAL_THIRTY (A));                SIM05630
PUT FILE(SYSPRINT) EDIT ('YBAR AFTER READIT = ',Y_BAR) (SKIP,A,B(30));
   OMT: END READ_IT;                                                        SIM05640
                                                                            SIM05650
REV_OCTAL: PROCEDURE (BIN) BIT(3);                                          SIM05660
PUT FILE(SYSPRINT) EDIT ('IN PROC REV OCTAL ')(SKIP,A);
            DCL BIN CHAR (1);                                               SIM05670
            IF BIN = '0' THEN RETURN ('000'B);                              SIM05680
            IF BIN = '1' THEN RETURN ('001'B);                              SIM05690
            IF BIN = '2' THEN RETURN ('010'B);                              SIM05700
            IF BIN = '3' THEN RETURN ('011'B);                              SIM05710
            IF BIN = '4' THEN RETURN ('100'B);                              SIM05720
            IF BIN = '5' THEN RETURN ('101'B);                              SIM05730
            IF BIN = '6' THEN RETURN ('110'B);                              SIM05740
            IF BIN = '7' THEN RETURN ('111'B);                              SIM05750
END REV_OCTAL;                                                              SIM05760
                                                                            SIM05770
                                                                            SIM05780
REP_STO: PROCEDURE (EINS, ZWEI);                                            SIM05790
PUT FILE(SYSPRINT) EDIT ('IN PROC REP STO   ')(SKIP,A);
          DCL EINS BIT (30);                                                SIM05800
          DCL ZWEI BIT (15);                                                SIM05810
          DCL VALUE FIXED BINARY (15);                                      SIM05820
          IF (K = 0 | K = 4 | K = 7) THEN RETURN;                           SIM05830
          VALUE = VAL_15 (ZWEI);                                            SIM05840
          IF K = 1 | K = 5 THEN IF VALUE > 32767 THEN
```

```
          SUBSTR (MEM2 (VALUE - 32767),16,15) =
          SUBSTR (EINS,16,15);
          ELSE SUBSTR (MEM1 (VALUE),16,15) = SUBSTR (EINS,16,15);
          IF K = 2 | K = 6 THEN IF VALUE > 32767 THEN
          SUBSTR (MEM2 (VALUE - 32767),1,15) =
          SUBSTR (EINS,16,15);
          ELSE SUBSTR (MEM1 (VALUE),1,15) = SUBSTR (EINS,16,15);
          IF K = 3 THEN IF VALUE > 32767 THEN
          MEM2 (VALUE - 32767) = EINS;
          ELSE MEM1 (VALUE ) = EINS;
END REP_STO;                                                          SIM05860
                                                                      SIM05870
REP_READ: PROCEDURE;                                                  SIM05880
PUT FILE(SYSPRINT) EDIT ('IN PROC REP READ  ')(SKIP,A);
          K_DEC = OCTAL (K);
          CALL REP_STR ('0',15);
          IF K_DEC = 1 THEN
               Y_BAR = ASTR || SUBSTR (MEMORY (VAL_15 (Y)),16,15);
          IF K_DEC = 2 THEN
               Y_BAR = ASTR || SUBSTR (MEMORY (VAL_15 (Y)),1,15);
          IF K_DEC = 3 THEN
               Y_BAR = MEMORY (VAL_15 (Y));
          CALL REP_STR (SUBSTR (Y_BAR,17,1),15);
          IF K_DEC = 5 THEN
          Y_BAR = ASTR || SUBSTR (MEMORY (VAL_15 (Y)),16,15);
          CALL REP_STR (SUBSTR (Y_BAR,1,1),15);
          IF K_DEC = 6 THEN
          Y_BAR = ASTR || SUBSTR (MEMORY (VAL_15 (Y)),1,15);
END REP_READ;                                                         SIM05900
                                                                      SIM05910
J_CHANGE: PROCEDURE;                                                  SIM05920
PUT FILE(SYSPRINT) EDIT ('IN PROC J CHANGE  ')(SKIP,A);
           J_DEC = OCTAL (J);                                         SIM05940
            IF J_DEC = 0 THEN RETURN;                                 SIM05950
            IF J_DEC = 1 THEN GO TO SKIP2;                            SIM05960
            IF J_DEC = 2 THEN IF Q > -1 THEN GO TO SKIP2;             SIM05970
                          ELSE RETURN;                                SIM05980
            IF J_DEC = 3 THEN IF Q < 0 THEN GO TO SKIP2;              SIM05990
                          ELSE RETURN;                                SIM06000
            IF J_DEC = 4 THEN IF A = 0 THEN GO TO SKIP2;              SIM06010
                          ELSE RETURN;                                SIM06020
            IF J_DEC = 5 THEN IF A ¬= 0 THEN GO TO SKIP2;             SIM06030
                          ELSE RETURN;                                SIM06040
            IF J_DEC = 6 THEN IF A > -1 THEN GO TO SKIP2;             SIM06050
                          ELSE RETURN;                                SIM06060
            IF J_DEC = 7 THEN IF A < 0 THEN GO TO SKIP2;              SIM06070
                          ELSE RETURN;                                SIM06080
      SKIP2: P_DEC = P_DEC + 1;                                       SIM06090
```

```
            P = BIT (P_DEC,15);                                          SIM06100
          END J_CHANGE;                                                  SIM06110
                                                                         SIM06120
                                                                         TIM01910
SPEC_J: PROCEDURE;                                                       TIM01920
        DCL J_DEC FIXED BINARY (15);                                     TIM01930
        J_DEC = OCTAL (J);                                               TIM01940
        IF J_DEC = 1                    |                                TIM01950
           J_DEC = 2  & A > 0           |                                TIM01960
           J_DEC = 3   & A < 0          |                                TIM01970
           J_DEC = 4   & Q = 0          |                                TIM01980
           J_DEC = 5   & Q ¬= 0         |                                TIM01990
           J_DEC = 6   & Q > 0          |                                TIM02000
           J_DEC = 7   & Q < 0    THEN DO;                              TIM02010
               P_DEC = P_DEC + 1;                                        TIM02020
               P = BIT (P_DEC,15);                                       TIM02030
               END;                                                      TIM02040
      END SPEC_J;                                                        TIM02050
                                                                         TIM02060
PAR_CHECK: PROCEDURE (CHECK) CHAR (4);                                   SIM06130
PUT FILE(SYSPRINT) EDIT ('IN PROC PAR CHECK ')(SKIP,A);
          DCL CHECK BIT (30);                                            SIM06140
          DCL FINAL CHAR (4);                                            SIM06150
          DCL FLAG FIXED BINARY (15) INITIAL (0);                        SIM06160
          DCL COUNTER FIXED BINARY (15);                                 SIM06170
          DO COUNTER = 1 TO 30;                                          SIM06180
             IF SUBSTR (CHECK,COUNTER,1) = '1'B THEN IF FLAG = 1 THEN    SIM06190
                                                  FLAG = 0;              SIM06200
                                ELSE FLAG = 1;                           SIM06210
             END;                                                        SIM06220
    IF FLAG = 0 THEN FINAL = 'EVEN';
                ELSE FINAL = 'ODD ';                                     SIM06240
          RETURN (FINAL);                                                SIM06250
END PAR_CHECK;                                                           SIM06260
                                                                         SIM06270
MEMORY: PROCEDURE (ALPHA) BIT (30);                                      SIM06280
  PUT FILE (SYSPRINT) EDIT ('IN PROC MEMORY ALPHA IS ',ALPHA)
                          (SKIP,A,F(8));
        DCL ALPHA FIXED BINARY (15);                                     SIM06290
          IF ALPHA > 65536 THEN RETURN (MEM1(0));                        LES00010
        IF ALPHA > 32767 THEN RETURN (MEM2 (ALPHA - 13516));             SIM06300
        ELSE RETURN (MEM1 (ALPHA));                                      SIM06310
        END MEMORY;                                                      SIM06320
                                                                         SIM06330
                                                                         SIM06340
   FINISH: END SIMUL;                                                    LES00390
```

```
MASTER: PROC OPTIONS (MAIN);                                          RAY00020
DCL INPUT CHAR (80) VAR;                                              RAY00030
DCL BUF CHAR (80) VAR;                                                RAY00040
DCL CODE BIT (30);                                                    RAY00050
DCL LAB FIXED BINARY (15);                                            RAY00060
DCL MACH (1000) BIT (30);                                             RAY00070
DCL FORMAT FIXED BINARY (15);                                         RAY00080
DCL SYSPRINT FILE STREAM PRINT ENV (F(133));                          RAY00090
DCL OP CHAR (3);                                                      RAY00100
DCL JDEC FIXED BINARY (15);                                           RAY00110
DCL J BIT (3) VAR;                                                    RAY00120
DCL J_HAT BIT (4) VAR;                                                RAY00130
DCL K BIT (3);                                                        RAY00140
DCL K_HAT BIT (2);                                                    RAY00150
DCL OPCODE BIT (6);                                                   RAY00160
DCL Y BIT (15);                                                       RAY00170
DCL Y_1 BIT (9);
DCL Y_2 BIT (17);
DCL B BIT (3);                                                        RAY00180
DCL SP_77 BIT (6);                                                    RAY00190
DCL WORD (2) CHAR (5) VAR;                                            RAY00200
DCL WORDM CHAR (5);                                                   RAY00210
DCL (M,N,O) FIXED BIN (15);                                           RAY00220
DCL CNT FIXED BIN (15) INITIAL (1);                                   RAY00230
DCL FORM_NO ENTRY (FIXED BINARY (15));                                RAY00240
DCL VAR (48) LABEL;                                                   RAY00250
DCL TYPE CHAR (144);                                                  RAY00260
          TYPE = 'ADDSUBMULDIVEXPLESEQLLEQNEQGEQGTRNEG' ||            RAY00270
                 'NOTANDBOPONETWOLODSTODELSTDXITBFNBFC' ||            RAY00280
                 'BBCBRSBSCNOPPRORTNGETRETRDVWRVWRSDMP' ||            RAY00290
                 'TABROWSCRSAVUNSSUPLITEOFDELDUPXCH048';              RAY00300
DCL REV_OCT ENTRY (CHAR (5) VAR) RETURNS (BIT (15));                  RAY00310
DCL REV ENTRY (CHAR(1)) RETURNS (BIT (3));                            RAY00320
START: M = 1;                                                         RAY00330
SP_77 = '111111';                                                     RAY00340
DISPLAY ('I AM READY TO WORK') REPLY (BUF);                           RAY00350
    DO N = 1 TO 2;                                                    RAY00360
    WORD (N) = '';                                                    RAY00370
    END;                                                              RAY00380
   /*     THIS SECTION CHECKS FOR A LABEL INPUT,                      RAY00390
           ASCERTAINS THE LABEL NUMBER AND PUTS THE                   RAY00400
           PROPER ADDRESS OF THE PRESENT INSTRUCTION INTO             RAY00410
           THE MEMORY LOCATION OF THE LABEL NUMBER.       */          RAY00420
                                                                      RAY00430
     IF SUBSTR (BUF,1,1) = '5' THEN DO;    /* LABELS BEGIN WITH 5 */  RAY00440
          LAB = REV_OCT ('00' || SUBSTR (BUF,1,3));                   RAY00450
          BUF = SUBSTR (BUF,6);          /* REMOVES LABEL FROM BUFFER */ RAY00460
           K = '000';                                                 RAY00470
```

```
            J = '000';                                                    RAY00480
            B = '000';                                                    RAY00490
            Y = BIT (CNT+2,15);      /* Y HAS PRESENT INST ADDRESS */     RAY00500
            OPCODE = '001001'B;                                           RAY00510
            CALL FORM_NO (1);       /* GENERATE MACHINE CODE TO LOAD A */RAY00520
            K = '011';                                                    RAY00530
            OPCODE = '001101'B;                                           RAY00540
            Y = BIT (LAB,15);  /* Y HAS LABEL NUMBER */                   RAY00550
            J = '000'B;                                                   RAY00560
            B = '000'B;                                                   RAY00570
            CALL FORM_NO (1);   /* GENERATE MACHINE CODE TO STORE A */    RAY00580
          END;                                                            RAY00590
AGAIN: IF SUBSTR (BUF,1,1) ¬= ',' THEN                                    RAY00600
            WORD (M) = WORD (M) || SUBSTR (BUF,1,1);                      RAY00610
ELSE M = M + 1;                                                           RAY00620
BUF = SUBSTR (BUF,2);                                                     RAY00630
IF SUBSTR (BUF,1,2) = '  ' THEN GO TO COMPUTE;                           RAY00640
GO TO AGAIN;                                                              RAY00650
COMPUTE: OP = WORD (1);                                                   RAY00660
JDEC = TRUNC (INDEX (TYPE,OP) / 3) + 1;                                   RAY00670
PUT FILE (SYSPRINT) EDIT (' JDEC IS ', JDEC) (SKIP,A,F(6));               RAY00680
GO TO VAR (JDEC);                                                         RAY00690
                                                                          RAY00700
VAR (1): PUT FILE (SYSPRINT) EDIT ('ADD') (SKIP,A);                       RAY00710
         OPCODE = '010000'B;                                              RAY00720
         K = '000'B;                                                      RAY00730
         J = '000'B;                                                      RAY00740
         B = '000'B;                                                      RAY00750
         FORMAT = 1;                                                      RAY00760
         Y = REV_OCT (WORD (2));                                          RAY00770
PUT FILE (SYSPRINT) EDIT ('Y AFTER REV OCT IN IS ',Y) (SKIP,A,F(6));      RAY00780
         CALL FORM_NO (FORMAT);                                           RAY00790
         GO TO ENDING;                                                    RAY00800
                                                                          RAY00810
VAR (2): PUT FILE (SYSPRINT) EDIT ('SUB') (SKIP,A);                       RAY00820
          OPCODE = '010001';                                              RAY00830
          K = '000';                                                      RAY00840
          J = '000';                                                      RAY00850
          B = '000';                                                      RAY00860
          FORMAT = 1;                                                     RAY00870
          Y = REV_OCT (WORD (2));                                         RAY00880
PUT FILE (SYSPRINT) EDIT (' Y AFTER REV OCT IN  IS ',Y) (SKIP,A,F(6));    RAY00890
          CALL FORM_NO (FORMAT);                                          RAY00900
         GO TO ENDING;                                                    RAY00910
                                                                          RAY00920
VAR (3): PUT FILE (SYSPRINT) EDIT ('MUL') (SKIP,A);                       RAY00930
          K = '000';                                                      RAY00940
          B = '000';                                                      RAY00950
```

```
          J = '000';                                                    RAY00960
          OPCODE = '001101';                                            RAY00970
          CALL FORM_NO (1);                                             RAY00980
          OPCODE = '010010';                                            RAY00990
          Y = REV_OCT (WORD (2));                                       RAY01000
          CALL FORM_NO (1);                                             RAY01010
          OPCODE = '000111';                                            RAY01020
          SUBSTR (Y,10,6) = '011000';                                   RAY01030
          CALL FORM_NO (1);                                             RAY01040
          GO TO ENDING;                                                 RAY01050
                                                                        RAY01060
VAR (4): PUT FILE (SYSPRINT) EDIT ('DIV') (SKIP,A);                     RAY01070
          K = '000';                                                    RAY01080
          J = '000';                                                    RAY01090
          B = '000';                                                    RAY01100
                                                                        RAY01110
          OPCODE = '001101';                                            RAY01120
          CALL FORM_NO (1);                                             RAY01130
          Y = '000000000000000';                                        RAY01140
          OPCODE = '001001';                                            RAY01150
          CALL FORM_NO (1);                                             RAY01160
          Y = REV_OCT (WORD (2));                                       RAY01170
          OPCODE = '010011';                                            RAY01180
          CALL FORM_NO (1);                                             RAY01190
          SUBSTR (Y,10,6) = '011000';                                   RAY01200
          OPCODE = '000111';                                            RAY01210
          CALL FORM_NO (1);                                             RAY01220
          GO TO ENDING;                                                 RAY01230
                                                                        RAY01240
VAR (5): PUT FILE (SYSPRINT) EDIT ('EXP') (SKIP,A);                     RAY01250
                                                                        RAY01260
          J = '000';                                                    RAY01270
          K = '000';                                                    RAY01280
          B = '000';                                                    RAY01290
          OPCODE = '001101';                                            RAY01300
          CALL FORM_NO (1);                                             RAY01310
          Y = '000000001110010';                                        RAY01320
          CALL FORM_NO (1);                                             RAY01330
          M = REV_OCT (WORD (2));                                       RAY01340
           OPCODE = '010010';                                           RAY01350
          DO WHILE (M ¬= 0);                                            RAY01360
          CALL FORM_NO (1);                                             RAY01370
          M = M - 1;                                                    RAY01380
          END;                                                          RAY01390
          OPCODE = '000111';                                            RAY01400
          SUBSTR (Y,10,6) = '011000';                                   RAY01410
          CALL FORM_NO (1);                                             RAY01420
          GO TO ENDING;                                                 RAY01430
```

```
VAR (6): PUT FILE (SYSPRINT) EDIT ('LES') (SKIP,A);                    RAY01440
          K = '000'B;                                                  RAY01450
          J = '000'B;                                                  RAY01460
          B = '000'B;                                                  RAY01470
          Y = '0000000000000000'B;                                     RAY01480
          OPCODE = '001101'B;                                          RAY01490
          CALL FORM_NO (1);                    /* SHIFT A TO Q */      RAY01500
          Y = '0000000000000001'B;                                     RAY01510
          OPCODE = '010111'B;                                          RAY01520
          CALL FORM_NO (1);              /* SUBTRACT 1 FROM Q */       RAY01530
          J = '010'B;                                                  RAY01540
          Y = REV_OCT (WORD (2));                                      RAY01550
          OPCODE = '000100'B;                                          RAY01560
          CALL FORM_NO (1);        /* COMPARE X WITH Q FOR SKIP */     RAY01570
         GO TO ENDING;                                                 RAY01580
                                                                       RAY01590
VAR (7): PUT FILE (SYSPRINT) EDIT ('EQL') (SKIP,A);                    RAY01600
                                                                       RAY01610
         GO TO ENDING;                                                 RAY01620
                                                                       RAY01630
VAR (8): PUT FILE (SYSPRINT) EDIT ('LEQ') (SKIP,A);                    RAY01640
          J = '110'B;                                                  RAY01650
          K = '000'B;                                                  RAY01660
          B = '000'B;                                                  RAY01670
          Y = REV_OCT (WORD(2));                                       RAY01680
          OPCODE = '000100'B;                                          RAY01690
          CALL FORM_NO (1);         /* COMPARE X WITH A FOR SKIP */    RAY01700
        GO TO ENDING;                                                  RAY01710
                                                                       RAY01720
VAR (9): PUT FILE (SYSPRINT) EDIT ('NEQ') (SKIP,A);                    RAY01730
        GO TO ENDING;                                                  RAY01740
                                                                       RAY01750
VAR (10): PUT FILE (SYSPRINT) EDIT ('GEQ') (SKIP,A);                   RAY01760
                                                                       RAY01770
         GO TO ENDING;                                                 RAY01780
                                                                       RAY01790
VAR (11): PUT FILE (SYSPRINT) EDIT ('GTR') (SKIP,A);                   RAY01800
          J = '111'B;                                                  RAY01810
          K = '000'B;                                                  RAY01820
          B = '000'B;                                                  RAY01830
          Y = REV_OCT (WORD (2));                                      RAY01840
          OPCODE = '000100'B;                                          RAY01850
          CALL FORM_NO (1);       /* COMPARE X WITH A FRO SKIP */      RAY01860
          GO TO ENDING;                                                RAY01870
                                                                       RAY01880
VAR (12): PUT FILE (SYSPRINT) EDIT ('NEG') (SKIP,A);                   RAY01890
          GO TO ENDING;                                                RAY01900
                                                                       RAY01910
```

```
                                                                        RAY01920
VAR (13): PUT FILE (SYSPRINT) EDIT ('NOT') (SKIP,A);                    RAY01930
          J = '000'B;                                                   RAY01940
          K = '100'B;                                                   RAY01950
          B = '000'B;                                                   RAY01960
          Y = '0000000000000000'B;                                      RAY01970
          OPCODE = '001101'B;                                           RAY01980
          CALL FORM_NO (1);          /* COMPLIMENT A */                 RAY01990
          GO TO ENDING;                                                 RAY02000
                                                                        RAY02010
VAR (14): PUT FILE (SYSPRINT) EDIT ('AND') (SKIP,A);                    RAY02020
K = '000'B;
B = '000'B;
J = '110'B;
Y = REV_OCT (WORD (2));
OPCODE = '000100'B;
CALL FORM_NO (1);     /*  TEST A = X  */
J = '000'B;
K = '100'B;
Y = '1111111111111111'B;
OPCODE = '101001'B;
CALL FORM_NO (1);    /* COMPLIMENT  */
          GO TO ENDING;                                                 RAY02030
                                                                        RAY02040
VAR (15): PUT FILE (SYSPRINT) EDIT ('BOR') (SKIP,A);                    RAY02050
B = '000'B;
J = '000'B;
K = '100'B;
Y = '1111111111111111'B;
OPCODE = '101001'B;
CALL FORM_NO (1);       /* COMPLIMENT  */
          GO TO ENDING;                                                 RAY02060
                                                                        RAY02070
VAR (16): PUT FILE (SYSPRINT) EDIT ('ONE') (SKIP,A);                    RAY02080
          B = '000'B;                                                   RAY02090
          J = '000'B;                                                   RAY02100
          K = '000'B;                                                   RAY02110
          Y = REV_OCT (WORD (2));                                       RAY02120
          OPCODE = '001001'B;                                           RAY02130
          CALL FORM_NO (1);           /* ENTER X INTO A */              RAY02140
          GO TO ENDING;                                                 RAY02150
                                                                        RAY02160
VAR (17): PUT FILE (SYSPRINT) EDIT ('TWO') (SKIP,A);                    RAY02170
          B = '000';                                                    RAY02180
          J = '000';                                                    RAY02190
          K = '000';                                                    RAY02200
          Y = REV_OCT (WORD (2));                                       RAY02210
          OPCODE = '001001';                                            RAY02220
```

```
            CALL FORM_NO (1);                                          RAY02230
            Y = REV_OCT (WORD (3));                                    RAY02240
            OPCODE = '001000';                                         RAY02250
            CALL FORM_NO (1);                                          RAY02260
            GO TO ENDING;                                              RAY02270
                                                                       RAY02280
VAR (18):   PUT FILE (SYSPRINT) EDIT ('LQD') (SKIP,A);                RAY02290
            B = '000';                                                 RAY02300
            J = '000';                                                 RAY02310
            K = '000';                                                 RAY02320
            Y = REV_OCT (WORD (2));                                    RAY02330
            OPCODE = '001001';                                         RAY02340
            CALL FORM_NO (1);                                          RAY02350
            GO TO ENDING;                                              RAY02360
                                                                       RAY02370
VAR (19):   PUT FILE (SYSPRINT) EDIT ('STO') (SKIP,A);                RAY02380
            B = '000';                                                 RAY02390
            J = '000';                                                 RAY02400
            K = '011';                                                 RAY02410
            Y = REV_OCT (WORD (2));                                    RAY02420
            OPCODE = '001101';                                         RAY02430
            CALL FORM_NO (1);                                          RAY02440
            GO TO ENDING;                                              RAY02450
                                                                       RAY02460
VAR (20):   PUT FILE (SYSPRINT) EDIT ('DEL') (SKIP,A);                RAY02470
B = '000'B;
J = '000'B;
K = '000'B;
Y = '0000000000000000'B;
OPCODE = '001001'B;
CALL FORM_NO (1);              /*  ZERO OUT A    */
            GO TO ENDING;                                              RAY02480
                                                                       RAY02490
VAR (21):   PUT FILE (SYSPRINT) EDIT ('STD') (SKIP,A);                RAY02500
K = '011'B;
Y = REV_OCT (WORD (2));
OPCODE = '001101'B;
CALL FORM_NO (1);    /*  STORE A IN ADDRESS X  */
Y = '0000000000000000'B;
OPCODE = '001001'B;
K = '000'B;
CALL FORM_NO (1);       /*  ZERO OUT A  */
            GO TO ENDING;                                              RAY02510
                                                                       RAY02520
VAR (22):   PUT FILE (SYSPRINT) EDIT ('XIT') (SKIP,A);                RAY02530
B = '000'B;
J = '000'B;
K = '000'B;
```

```
Y = '0000001100010000'B;  /* BRANCH TO IPL POINT  */
OPCODE = '110001'B;
CALL FORM_NO (1);      /*   JUMP TO 620  */
          GO TO ENDING;                                                  RAY02540
                                                                         RAY02550
VAR (23): PUT FILE (SYSPRINT) EDIT ('BFN') (SKIP,A);                     RAY02560
          GO TO ENDING;                                                  RAY02570
                                                                         RAY02580
VAR (24): PUT FILE (SYSPRINT) EDIT ('BFC') (SKIP,A);                     RAY02590
          GO TO ENDING;                                                  RAY02600
                                                                         RAY02610
VAR (25): PUT FILE (SYSPRINT) EDIT ('BBC') (SKIP,A);                     RAY02620
          GO TO ENDING;                                                  RAY02630
                                                                         RAY02640
VAR (26): PUT FILE (SYSPRINT) EDIT ('BRS') (SKIP,A);                     RAY02650
B = '000'B;
J = '001'B;
K = '000'B;
Y = REV_OCT (WORD (2));
OPCODE = '110000'B;
CALL FORM_NO (1);       /*   BRANCH TO  X  */
          GO TO ENDING;                                                  RAY02660
                                                                         RAY02670
VAR (27): PUT FILE (SYSPRINT) EDIT ('BSC') (SKIP,A);                     RAY02680
B = '000'B;
J = '101'B;
K = '000'B;
Y = REV_OCT (WORD (2));
OPCODE = '110000'B;
CALL FORM_NO (1);    /*  JUMP TO  X  */
          GO TO ENDING;                                                  RAY02690
                                                                         RAY02700
VAR (28): PUT FILE (SYSPRINT) EDIT ('NOP') (SKIP,A);                     RAY02710
B = '000'B;
J = '000'B;
K = '000'B;
Y = '000000000000000'B;
OPCODE = '001010'B;
CALL FORM_NO (1);     /*  ENTER B ZERO WITH ZERO  */
          GO TO ENDING;                                                  RAY02720
                                                                         RAY02730
VAR (29): PUT FILE (SYSPRINT) EDIT ('PRO') (SKIP,A);                     RAY02740
          GO TO ENDING;                                                  RAY02750
                                                                         RAY02760
VAR (30): PUT FILE (SYSPRINT) EDIT ('RTN') (SKIP,A);                     RAY02770
          GO TO ENDING;                                                  RAY02780
                                                                         RAY02790
VAR (31): PUT FILE (SYSPRINT) EDIT ('GET') (SKIP,A);                     RAY02800
```

```
          GO TO ENDING;                                                  RAY02810
                                                                         RAY02820
VAR (32): PUT FILE (SYSPRINT) EDIT ('RET') (SKIP,A);                     RAY02830
          GO TO ENDING;                                                  RAY02840
                                                                         RAY02850
VAR (33): PUT FILE (SYSPRINT) EDIT ('RDV') (SKIP,A);                     RAY02860
B = '000'B;
J = '000'B;
K = '000'B;
Y = '0000000001110010'B;
OPCODE = '001001'B;
CALL FORM_NO (1);    /*  ENTER 162 INTO A  */
Y = '0000000000001101'B;
OPCODE = '000110'B;
CALL FORM_NO (1);    /*  SHIFT A LEFT 15 BITS  */
K = '011'B;
Y = '0000000001000010'B;
OPCODE = '001101'B;
CALL FORM_NO (1);     /*  STORE A AT 102  */
J_HAT = '0010'B;
K_HAT = '00'B;
Y = '0000000001110010'B;
OPCODE = '111011'B;
CALL FORM_NO (1);    /*  INPUT VAR AT 162  */
K = '000'B;
Y = '0000000000000000'B;
OPCODE = '001001'B;
CALL FORM_NO (1);     /*   SET A = 0  */
Y = '0000000001110010'B;
OPCODE = '010100'B;
CALL FORM_NO (1);     /*  SET A = MEM ( 162)   */
          GO TO ENDING;                                                  RAY02870
                                                                         RAY02880
VAR (34): PUT FILE (SYSPRINT) EDIT ('WRV') (SKIP,A);                     RAY02890
B = '000'B;
J = '000'B;
K = '011'B;
Y = '0000000001110010'B;
OPCODE = '001101'B;
CALL FORM_NO (1);     /*   STORE A IN 162   */
K = '000'B;
Y = '0000000001110010'B;
OPCODE = '001001'B;
CALL FORM_NO (1);     /*  ENTER 162 INTO A   */
Y = '0000000000001101'B;
OPCODE = '000110'B;
CALL FORM_NO (1);   /*  SHIFT A LEFT 15 BITS   */
K = '011'B;
```

```
Y = '0000000010000010'B;
OPCODE = '001101'B;
CALL FORM_NO (1);     /*  STORE A AT 102  */
K_HAT = '00'B;
J_HAT = '0010'B;
OPCODE = '111011'B;
Y = '0000000001110010'B;
CALL FORM_NO (1);          /*  OUTPUT THE VARIABLE  */
          GO TO ENDING;                                          RAY02900
                                                                 RAY02910
VAR (35): PUT FILE (SYSPRINT) EDIT ('WRS') (SKIP,A);             RAY02920
B = '000'B;
J = '000'B;
K = '000'B;
OPCODE = '001001'B;
Y = REV_OCT (WORD (2));
CALL FORM_NO (1);    /*  ENTER X INTO A  */
Y = '0000000000000101'B;
OPCODE = '010000'B;
CALL FORM_NO (1);              /*  ADD 5 TO A   */
Y = '0000000000001101'B;
OPCODE = '000110'B;
CALL FORM_NO (1);    /*  SHIFT A LEFT 15 BITS  */
K = '011'B;
Y = '0000000010010010'B;
OPCODE = '001101'B;
Y = '0000000010000010'B;
CALL FORM_NO (1);     /*  STORE A AT 102   */
J_HAT = '0010'B;
K_HAT = '00'B;
OPCODE = '111011'B;
Y = '0000000000000101'B;
CALL FORM_NO (2);    /*   WRIT
          GO TO ENDING;                                          RAY02930
                                                                 RAY02940
VAR (36): PUT FILE (SYSPRINT) EDIT ('DMP') (SKIP,A);             RAY02950
          GO TO ENDING;                                          RAY02960
                                                                 RAY02970
VAR (37): PUT FILE (SYSPRINT) EDIT ('TAB') (SKIP,A);             RAY02980
          GO TO ENDING;                                          RAY02990
                                                                 RAY03000
VAR (38): PUT FILE (SYSPRINT) EDIT ('ROW') (SKIP,A);             RAY03010
          GO TO ENDING;                                          RAY03020
                                                                 RAY03030
VAR (39): PUT FILE (SYSPRINT) EDIT ('SCR') (SKIP,A);
          GO TO ENDING;                                          RAY03050
                                                                 RAY03060
VAR (40): PUT FILE (SYSPRINT) EDIT ('SAV') (SKIP,A);             RAY03070
```

```
            GO TO ENDING;                                                   RAY03080
                                                                            RAY03090
VAR (41):   PUT FILE (SYSPRINT) EDIT ('UNS') (SKIP,A);                     RAY03100
            GO TO ENDING;                                                   RAY03110
                                                                            RAY03120
VAR (42):   PUT FILE (SYSPRINT) EDIT ('SUP') (SKIP,A);                     RAY03130
            GO TO ENDING;                                                   RAY03140
                                                                            RAY03150
VAR (43):   PUT FILE (SYSPRINT) EDIT ('LIT') (SKIP,A);                     RAY03160
            B = '000'B;                                                     RAY03170
            J = '000'B;                                                     RAY03180
            K = '000'B;                                                     RAY03190
            Y = REV_OCT (WORD (2));                                         RAY03200
            OPCODE = '001001'B;                                             RAY03210
            CALL FORM_NO (1);          /* SET A = X */                      RAY03220
OPCODE = '001000'B;
CALL FORM_NO (1);    /*   SET Q = X    */
            GO TO ENDING;                                                   RAY03230
                                                                            RAY03240
VAR (44):   PUT FILE (SYSPRINT) EDIT ('EOF') (SKIP,A);                     RAY03250
            J = '100';                                                      RAY03260
            K = '000';                                                      RAY03270
            B = '000';                                                      RAY03280
            Y = '0000000000000000';                                         RAY03290
            OPCODE = '110001';                                              RAY03300
            CALL FORM_NO (1);                                               RAY03310
            GO TO ENDING;                                                   RAY03320
                                                                            RAY03330
VAR (45):   PUT FILE (SYSPRINT) EDIT ('DEL') (SKIP,A);                     RAY03340
            B = '000'B;                                                     RAY03350
            J = '000'B;                                                     RAY03360
            K = '000'B;                                                     RAY03370
            Y = '0000000000000000'B;                                        RAY03380
            OPCODE = '001001'B;                                             RAY03390
            CALL FORM_NO (1);              /* ENTER A 0 INTO A */           RAY03400
            GO TO ENDING;                                                   RAY03410
                                                                            RAY03420
VAR (46):   PUT FILE (SYSPRINT) EDIT ('DUP') (SKIP,A);                     RAY03430
            B = '000'B;                                                     RAY03440
            J = '000'B;                                                     RAY03450
            K = '000'B;                                                     RAY03460
            Y = '0000000000000000'B;                                        RAY03470
            OPCODE = '001101'B;                                             RAY03480
            CALL FORM_NO (1);            /* SET Q = A  */                   RAY03490
            GO TO ENDING;                                                   RAY03500
                                                                            RAY03510
VAR (47):   PUT FILE (SYSPRINT) EDIT ('XCH') (SKIP,A);                     RAY03520
            B = '000'B;                                                     RAY03530
```

```
          J = '000'B;                                                    RAY03540
          K = '000'B;                                                    RAY03550
          Y = '0000000000011000'B;                                       RAY03560
          OPCODE = '000111'B;                                            RAY03570
          CALL FORM_NO (1);          /* SHIFT AQ CIRCULARLY 30 BITS */   RAY03580
         GO TO ENDING;                                                   RAY03590
                                                                         RAY03600
VAR (48): PUT FILE (SYSPRINT) EDIT ('48') (SKIP,A);                      RAY03610
          GO TO ENDING;                                                  RAY03620
                                                                         RAY03630
                                                                         RAY03640
ENDING: DO O = 1 TO CNT - 1;                                             RAY03780
        PUT FILE (SYSPRINT)  EDIT ('MACHINE', O, ' IS ', MACH (O))       RAY03790
                    (SKIP,A,F(6),A,B(30));                               RAY03800
        END;                                                             RAY03810
        GO TO START;                                                     RAY03820

    /*   PROCEDURE FORM_NO ASSEMBLES THE VARIOUS FIELDS OF
         THE INSTRUCTION WORDS INTO THE PROPER FORMAT AS
         DESIGNATED BY THE PARAMETER FORM BY THE TYPE OF
         INSTRUCTION .                                          */

FORM_NO: PROCEDURE (FORM);                                               RAY03650
         DCL FORM FIXED BINARY (15);                                     RAY03660
         IF FORM = 1 THEN DO;                                            RAY03670
       PUT FILE (SYSPRINT) EDIT ('CNT IN FORMAT ',CNT) (SKIP,A,F(6));    RAY03680
PUT FILE (SYSPRINT) EDIT ('OPCODE IN FORMAT IS ',OPCODE) (SKIP,A,B(6));  RAY03690
PUT FILE (SYSPRINT) EDIT ('J IN FORMAT ONE IS ', J) (SKIP,A,B(3));       RAY03700
PUT FILE (SYSPRINT) EDIT ('K IN FORMAT ONE IS ', K) (SKIP,A,B(3));       RAY03710
PUT FILE (SYSPRINT) EDIT ('B IN FORMAT ONE IS ', B) (SKIP,A,B(3));       RAY03720
             MACH (CNT) = OPCODE || J || K || B || Y;                    RAY03730
         END;                                                            RAY03750
IF FORM = 2 THEN
         MACH (CNT) = OPCODE || J_HAT || K_HAT || B || Y;
IF FORM = 3 THEN
          MACH (CNT) = SP_77 || OPCODE ||B || Y;
IF FORM = 4 THEN
          MACH (CNT) = SP_77 || OPCODE || B || J_HAT || K_HAT || Y_1;
IF FORM = 5 THEN
          MACH (CNT) = SP_77 || OPCODE || '0'B || Y_2;
              CNT = CNT  + 1;                                            RAY03740
           END FORM_NO;                                                  RAY03760
                                                                         RAY03770
                                                                         RAY03830

    /*   PROCEDURE REV_OCT ACCEPTS THE PARAMETER OF A FIVE
         DIGIT CHARACTER STRING AND RETURNS THE THE OCTAL
         BIT REPRESENTATION FOR THE STRING.                     */
```

```
REV_OCT: PROCEDURE (VALUE) BIT (15);                                    RAY03840
         DCL VALUE CHAR (5);                                            RAY03850
         DCL MM FIXED BINARY (15);                                      RAY03860
         DCL RET BIT (15) VAR INITIAL ('');                             RAY03870
         DO MM = 1 TO 5;                                                RAY03880
           RET = RET || REV (SUBSTR (VALUE,MM,1));                      RAY03890
         END;                                                           RAY03900
         RETURN (RET);                                                  RAY03910
         END REV_OCT;                                                   RAY03920
                                                                        RAY03930
    /*   PROCEDURE REV RETURNS THE OCTAL BIT REPRESENTATION
         OF THE SINGLE NUMERIC CHARACTER INPUT.                    */

REV: PROCEDURE (ONECHAR) BIT (3);                                       RAY03940
     DCL ONECHAR CHAR (1);                                              RAY03950
     DCL THING BIT (3);                                                 RAY03960
     IF ONECHAR = '0' THEN RETURN ('000'B);                             RAY03970
     IF ONECHAR = '1' THEN RETURN ('001'B);                             RAY03980
     IF ONECHAR = '2' THEN RETURN ('010'B);                             RAY03990
     IF ONECHAR = '3' THEN RETURN ('011'B);                             RAY04000
     IF ONECHAR = '4' THEN RETURN ('100'B);                             RAY04010
     IF ONECHAR = '5' THEN RETURN ('101'B);                             RAY04020
     IF ONECHAR = '6' THEN RETURN ('110'B);                             RAY04030
     ELSE RETURN ('111'B);                                              RAY04040
     END REV;                                                           RAY04050
END MASTER;                                                             RAY04060
                                                                        RAY04070
```

BIBLIOGRAPHY

1. Sperry Rand Corporation, UNIVAC 1830-A Computer Technical Description.

2. Kildall, G. A., The Balgol-2 Programming System, Naval Postgraduate School, 1970.

3. McKeeman, W. M., Horning, J. J., and Wortman, D. B., A Compiler Generator, Prentice-Hall, Inc., 1970.

4. Orlicky, J., The Successful Computer System, McGraw Hill Book Company, 1969.

5. IBM System/360 PL/I Reference Manual, International Business Machine Company, 1968.

6. P-3C ORION Flight Crew Training, Volume I, Department of the Navy 12ND NASMF P1543 (VP-31), 1970.

7. P-3C ORION Flight Crew Training, Volume II, Department of the Navy 12ND NASMF P1543 (VP-31), 1970.

8. CP 901 ASQ-114 Computer Word Formats, Department of the Navy P-208-015-0, 1969.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Asst Professor G. H. Syms Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
4.	Lt. G. A. Kildall, USN Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
5.	Lt. Dennis E. Ray, USN 246 Harlem Road Pasadena, Maryland 21122	1
6.	LCDR E. A. Singer, USN ASW Systems Project Office Code ASW 251 Navy Department Washington, D. C. 20360	1
7.	Commanding Officer VP-31 Attn: LCDR Grant Fuller, USN NAS Moffett Field Mountain View, California	3

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

ORIGINATING ACTIVITY (Corporate author)
Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

REPORT TITLE
The Design, Development and Translation of General Purpose Software for the P3C Aircraft's Digital Computer

DESCRIPTIVE NOTES (Type of report and inclusive dates)
Master's Thesis; December 1971

AUTHOR(S) (First name, middle initial, last name)
Dennis Edward Ray

REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
December 1971	91	8

CONTRACT OR GRANT NO.

PROJECT NO.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

DISTRIBUTION STATEMENT
Approved for public release; distribution unlimited.

SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Naval Postgraduate School
Monterey, California

ABSTRACT

The Navy has incorporated a modified UNIVAC 1830-A (CP 901 or ASQ/114) "mini" digital computer into its P3C aircraft. This ASQ/114 computer system is presently used only during aircraft testing and flying. In the near future, fifty or more of these digital systems will be operational and will sit virtually idle about 40% of the time. Hence, this project was undertaken to improve the computer utilization, and to provide the individual squadrons with an administrative computer capability.

Six specific tasks leading to the implementation of a general purpose operating system have been undertaken: a feasibility study, development of a CP 901 translator, design and development of an assembler, design study of the bootstrapping technique, design of a FORTRAN compiler, and the design of a control operating system.

The documentation of these six tasks is intended to aid in the development of the final system.

DD FORM 1 NOV 65 1473 (PAGE 1)
S/N 0101-807-6811

Security Classification

A-31408

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
P3C						
General Purpose Software						
Digital Computer						

DRM
ov 65 1473 (BACK)

807-6821